Maricica Nistor

Network Coding Protocols with Delay and Energy Constraints

MAP-tele Doctoral Programme em Telecommunications

Departamento de Engenharia Eletrotécnica e de Computadores
Faculdade de Engenharia da Universidade do Porto

May 2014

Maricica Nistor

Network Coding Protocols with Delay and Energy Constraints

Tese submetida à Faculdade de Engenharia da Universidade do Porto para obtenção do grau de Doutor em Telecomunicações

Orientadores:
Professor Doutor João Barros, Universidade do Porto, Portugal
Professor Doutor Daniel E. Lucani, Aalborg University, Denmark

May 2014

Acknowledgements

I would like to thank my advisor, Professor João Barros, for his support, encouragement, and guidance throughout my PhD. I would also like to thank my co-advisor, Professor Daniel E. Lucani, for his help and patience, and the way he really made my life more enjoyable during these last years.

I thank my colleagues from IT Porto for the good moments we spent together and all research collaborators I worked with during my PhD.

A special thank goes to my family for the unconditional support, trust, and because they never stop to believe in me. I would also like to thank my friends for the encouragements, advises, and for being always close to me.

Finally, I would like to thank Luis Monteiro, for his love, patience, and support. I dedicate this work to Luis Monteiro!

This work was partially funded by the Fundação Calouste Gulbenkian under grant 103905 and by Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology) under grant SFRH-BD-61953-2009.

Abstract

Network coding is a technique that proposes a different approach for the protocol design in data communication networks. Thus, the nodes in the network are allowed not only to store and forward data packets, but also to process and mix different packets in a single coded packet. By using this technique, the throughput and robustness of the network can be significantly improved. However, the transmission delay of network coding is still not well understood. In real-time communication systems with stringent delay constraints, understanding the transmission delay distribution is at the core of implementing network coding in practical scenarios. Moreover, the benefits of network coding for broadcast scenarios have been proven, but the use of this technique in data gathering applications is limited. Unlike broadcast applications, where the main objective is to minimize the transmission delay, in data gathering applications the challenge is to reduce the data collection time, called the completion time. The inherent diversity and requirements of data gathering applications make this completion time minimization even more challenging and application-driven. Beyond transmission time considerations, network coding requires additional processing to generate the coded packets, which may influence the data transmission performance. In practical scenarios, where the power levels are fixed for different processes, e.g., transmission of data, generation of coded packets, the main challenge to improve the energy efficiency of the system is to decrease the transmission time. This implies careful attention in terms of energy efficiency of the overall system.

Thus, the focus of the thesis is on the design of communication protocols based on network coding subject to performance metrics that include the data transmission time, the data collection time, and the energy consumption. Moreover, we consider different constraints, i.e., application constraints (e.g., deadline, network size, topology, length of the data messages), network constraints (e.g., data loss), design constraints (e.g., feedback type, limited number of data packets).

We start by exploring the delay behaviour of network coding in broadcast applications. Previous results are typically asymptotic in nature or focus

mainly on the average delay performance. Seeking to characterize the complete delay distribution of network coding, we present a methodology that is feasible for fixed number of receivers, limited field size and number of data messages (generation size). Our findings can be used to optimize network coding protocols with respect to not only to their average but also to their worst-case delay.

Second, we deal with the data transmission performance for data gathering applications, where the broadcast properties of network coding can be again explored in more challenged networks. Here, we employ different sparsity levels for the coded data messages and types of feedback, and analyse the benefits of network coding for two topologies, i.e., random and line networks, using analytical and numerical results. The findings can be extended to large scale networks. We conclude that minimizing the completion time can be achieved for a wide variety of sparsity-feedback pairs given the opportunity that network designers have to choose the setting that best matches their devices capabilities.

We then focus on the hardware characteristics of battery-powered devices that are critical for the design of energy efficient communication protocols. Aiming at a systematic approach to choose protocols based on specific devices or to devise protocols that adapt to device characteristics, we first identify the hardware features of various devices and define a total energy consumption model of the network, which leads to a new hardware abstraction. Secondly, we illustrate the need of the hardware abstraction by evaluating the energy cost of two communication protocols, one based on network coding and the other based on Automatic Repeat reQuest and Time Division Multiple Access. The findings show that (a) the energy cost for a protocol varies significantly on different device platforms and (b) the protocols can be adapted to the underlying hardware for a maximum energy efficiency. Our results are supported with real-life measurements using sensor motes, i.e., TelosB.

With this in mind, we use the new hardware abstraction to provide mechanisms to optimize a communication protocol in terms of energy consumption. For this, we consider different levels of feedback to acknowledge the successful transmission of the data. Thus, we provide the potential of our contributions by illustrating how to (1) optimize the protocol to match the hardware and (2) optimize the protocol and the hardware jointly. The results show that the correct level of feedback is highly dependent on the platform of the device. We cross-validate the results with real-life implementations.

Resumo

A codificação de rede (*network coding*) é uma técnica que propõe uma abordagem different para a concepção de protocol em redes de comunicação de dados. Os nós na rede são permitidos não apenas para armazenar e transmitir pacotes de dados, mas também para processar e misturar diferentes pacotes num único pacote codificado. Ao utilizar esta técnica, o rendimento e a robustez da rede podem ser significativamente melhorados. No entanto, o atraso das transmissões (*transmission delay*) da codificação de rede ainda não é completamente compreendido. Em sistemas de comunicação em tempo real com restrições rígidas relativas ao atraso da informação, a compreensão da distribuição deste atraso é muito importante para a implementação de codificação de rede em cenários práticos. Os benefícios da codificação de rede em ambientes de transmissão em *broadcast* têm sido constantemente provados, porém, o uso desta técnica em aplicações de recolha massiva de dados é desconhecida. Ao contrario das aplicações de *broadcast* onde o principal objectivo é minimizar o atraso das transmissões, nas aplicações de recolha de dados o grande desafio é minimizar o tempo de coleta destes (*completion time*). A diversidade e os requisitos inerentes às aplicações de coleta de dados tornam esta minimização do tempo de coleta ainda mais desafiante e orientada às aplicações. Para além das considerações do tempo de transmissão, a codificação de rede requer processamento adicional para realizar as combinações dos pacotes de dados. Em cenários práticos, onde os níveis de potência são fixos para os diferentes processos, e.g., a transmissão de dados, geração de pacotes codificados, o principal desafio para melhorar a eficiência energética do sistema é diminuir o tempo de transmissão. Isto implica uma atenção especial em termos de eficiência energética do sistema global.

Assim, o foco da tese está na concepção de protocolos de comunicação baseados em codificação de rede, considerando como métricas de desempenho destes o tempo de transmissão, o tempo da coleta de dados e o consumo de energia. Além disso, tomamos em conta diferentes condicionalismos, como são os requisitos aplicacionais (e.g., o prazo de entrega, o tamanho da rede, a topologia, o comprimento das mensagens de dados), as limitações da rede

(e.g., perda de dados) e as condições impostas pelo projeto (e.g., o tipo de *feedback*, o número limitado dos pacotes de dados).

Em primeiro lugar, exploramos o comportamento de atraso de codificação de rede em aplicações de *broadcast*. Os resultados anteriores são tipicamente assintóticos e concentram-se principalmente no desempenho médio de atraso. Para determinar uma completa caracterização da distribuição de atraso para a codificação de rede, é apresentada uma metodologia que é valida para um determinado número de receptores, tamanho de campo e número de mensagens de dados (tamanho da geração). Os nossos resultados podem ser usados para optimizar os protocolos de codificação de rede não só em relação ao desempenho medio, mas também em função do comportamento no pior caso.

Em segundo lugar, lidamos com o desempenho da transmissão de dados em aplicações para a coleta dos mesmos, onde as propriedades de *broadcast* da codificação de rede podem ser novamente exploradas em redes mais constrangidas. Aqui, examinamos diferentes níveis de disperssão para as mensagens de dados codificados e tipos de *feedback*, ao mesmo tempo que analisamos os benefícios da codificação de rede para duas topologias, redes em linha (*line networks*) e aleatórias, utilizando métodos matemáticos e numéricos. Os resultados podem ser estendidos para redes em larga escala. Concluimos que a minimização do tempo da coleta de dados pode ser conseguida por uma grande variedade de pares de dispersão-*feedback*, dando liberdade aos *designers* de rede para escolher a melhor configuração que corresponde ás capacidades dos seus dispositivos.

Em seguida, concentramo-nos nas características de *hardware* de dispositivos alimentados por bateria, pois estas são cruciais para a eficiência energética dos protocolos de comunicação. Com o objetivo de escolher protocolos baseados em dispositivos específicos ou elaborar protocolos que se adaptam às características dos dispositivos, primeiro identificamos os recursos de *hardware* de vários dispositivos e definimos um modelo para o consumo total de energia da rede, o que leva à uma nova abstração de *hardware*. Em segundo lugar, mostramos a necessidade da abtração de *hardware* com uma avaliação do custo da energia de dois protocolos de comunicação, um baseado em codifição de rede e outro baseado em *Automatic Repeat reQuest* e Acesso Múltiplo por Divisão de Tempo (*Time Division Multiple Access*). Os resultados mostram que (a) o custo da energia para um protocolo varia significativamente em diferentes plataformas e (b) os protocolos podem ser adaptados para que o subjacente *hardware* tenha uma maior eficiência energética. Nossos resultados são compatíveis com medições reais extraidas de sensores do tipoenergia TelosB.

Nesta perspectiva, usamos a nova abstração de *hardware* para fornecer

mecanismos de modo a otimizar protocolos de comunicação em termos do consumo energético. Para isso, consideramos diferentes níveis de *feedback* para a confirmação de receção dos dados. As nossas contribuições mostram como (1) otimizar o protocolo para corresponder com o *hardware* e (2) otimizar o protocolo e o *hardware* em conjunto. Os resultados demonstram que o nível correcto de *feedback* é altamente dependente da plataforma do dispositivo. Validamos os resultados com implementações reais.

Contents

List of Tables

LIST OF TABLES

List of Figures

LIST OF FIGURES

List of Algorithms

Chapter 1

Introduction

The high energy consumption of the communication networks is increasingly becoming an important topic due to the continuous proliferation of more and more diverse networks [1, 2, 3, 4]. The main challenge is to design more energy efficient communication protocols. However, minimizing the energy consumption normally decreases the performance of the network in terms of throughput and delay. To meet the system requirements, we need to solve the trade-off between the performance of the network and energy consumption. On the one side, the performance of the network may be characterized by the data transmission time, i.e., the time to receive the data packets from a source node to one or more nodes in the network, or by the data completion time, i.e., the total time required to collect all the data from the nodes to a source node. Both of these metrics refer to delay and aim to be done in a timely manner and using a minimum of resources. On the other side, the energy consumption requires special attention in the design of the modern communication devices, particularly for battery-powered networks. Since the power consumption level of the communication devices is considered fixed, reducing the delay, e.g., the transmission time or the completion time, directly minimizes the energy consumption of the system. Hence, the communications engineer needs to consider from the start not only the system devices, but also how the communication protocols will affect the total energy consumption of it. In fact, the protocol designer needs to take into account the existence of several challenges imposed by (a) the application, e.g., different constraints in terms of the predefined deadlines, length of the data messages, topology, size of the network, (b) the network, e.g., unreliable channels, meaning the data message can be lost, and (c) the design, e.g., the processing operations, feedback type, and scheduling mechanisms. Such kind of approach is not what most of the state of the art does, as explained later in this chapter.

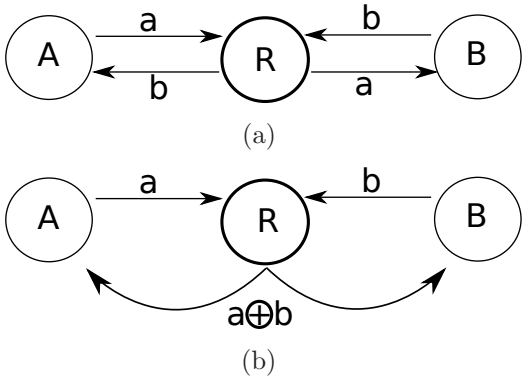


Figure 1.1: A typical wireless network coding example.

The more general question we want to answer is how to reduce the delay/the energy consumption of the network while maintaining the requirements of the application? With this in mind, the thesis is particularly focused on environments that are characterized by the mentioned constraints and cover relevant applications, such as wireless sensor networks (WSNs) and smart grids. Our performance metrics are the transmission delay, the completion time, and the energy consumption. We propose to design protocols that guarantee the delivery of the data before a specific deadline and meet the challenges of increasing the energy efficiency in communication networks.

The rest of the chapter is organized as follows. Section 1.1 introduces the network coding and presents the fundamentals of this technique, where Section 1.2 describes the main related works, revealing the principal research questions left open. This chapter concludes with Section 1.3, where the principal contributions of the thesis are presented.

1.1 Network Coding Fundamentals

Introduced for the first time in an information-theoretic setting by [5], network coding is necessary and sufficient to achieve the multicast capacity of a general network. The main idea of network coding is that data robustness and the throughput of the network can be considerably improved by allowing the intermediate nodes to perform algebraic combinations with the data flows. In particular, using the broadcast property of the wireless channel, network coding shows significant improvements from the traditional routing approach. The simplest example that demonstrates these benefits is provided in Figure 1.1. We consider a wireless broadcast network, where nodes A and B want to send a message to each other, but the radio range does not allow

them to communicate without a router R. In traditional 802.11, node A sends packet a to node R, which forwards it to node B, and B sends packet b to the router, which forwards it to node A. Thus, to exchange two packets, the traditional approach needs 4 time slots. Network coding achieves the same goal, but with fewer time slots. Node A and B send their packets to the router, one after the other; the router then XORs the two packets and broadcasts the XOR-ed version. Node A recovers the packet from B by XOR-ing again with its own, and node B recovers the packet from A in the same way. Thus, network coding reduces the number of time slots from 4 to 3. The free slot can be used to send new data, improving the wireless throughput.

In more sophisticated network coding protocols, the coded packets are generated by performing linear combinations of the original packets in a Galois Field $\mathbf{GF}(q)$. The algebraic framework in [6] and the emergence of Random Linear Network Coding (RLNC) [7, 8] led to practical applications. Here, network coding is performed in a fully distributed way by allowing the nodes to select their coefficients independently and uniformly at random from a Galois Field. The coefficients are appended to the header of each output packet. To recover the sent data, the destination waits until it receives enough independent packets to solve the resulting linear system by means of Gaussian elimination. Moreover, if the coding coefficients are chosen from a sufficiently large field size, the multicast capacity of the network can be achieved with high probability [9]. Hence, a receiver is not concerned any longer with the reception of a specific packet, but a new coded packet.

Two types of coding are possible, namely inter-flow and intra-flow network coding. The first one means that the packets used in combinations belong to different flows (example from Figure 1.1), e.g., [10]. It is particularly suitable for multi-hop wireless networks, where the packets from different sources are combined and then broadcasted together. A node can deliver multiple packets to different neighbours within a single transmission. On the other hand, intra-flow network coding implies that the packets belong from the same flow, e.g., [11]. It occurs when the encoder mixes its packets and broadcasts the obtained combination to the same destination, which might be one or several receivers. Each received packet contains some information about the original data packets. Thus, the source is only concerned with the transmission of a new combination and not a specific packet. When the destination receives enough coded packets to decode the original source packets, it can send an acknowledgement to the source to stop the transmission. The joint of these two approaches can be found in CORE [12].

Network coding has been demonstrated in practical scenarios. In order to reduce the number of transmitted packets, particularly in multiple receiver/transmitter scenarios, network coding proves high benefits, as [13]

shows. Additionally, MORE [11] presents a MAC-independent opportunistic routing protocol and demonstrates the throughput gains. This work also represents the first implementation of the intra-flow network coding. Further, [14] introduces the implementation of RLNC in practice, using mobile phones and laptops, and Galois field two. Here, the authors demonstrate that high encoding and decoding throughputs can be achieved. Moreover, CATWOMAN [15] shows the performance of network coding in a meshed network and tailored to commercial WiFi hardware. In video applications, [16] proposes an application for the Apple iPhone platform and shows the low bandwidth usage of this application. Additionally, the implementation of the CORE protocol [17] combines the benefits of intra-flow and inter-flow coding. The results reveal that the throughput follows an optimal trend and this scheme outperforms the COPE one from [13].

1.2 Previous Results and Open Research Questions

In classical network coding research, the network performance is optimized for throughput, which means to delivery the information data to a set of receivers within a minimum number of transmissions [18, 19, 20, 21]. But, delay is typically not a concern. Here, the receivers need to wait until sufficient data packets are received in order to be able to decode the original data packets. In this sense, the delay implications of the form of online network coding are explained in [22] and [23]. For a half duplex channel, [24] shows the optimal number of coded packets to be transmitted before the sender receives an acknowledgement packet. Moreover, [25] offers an efficient alternative for end-to-end reliability, which relies on feedback solely for the purpose of terminating the encoding process. In a line network, [26] proposes a characterization of delay for a limited number of transmitted packets through the network and the results show that the average delay is concentrated around its expectation. When feedback is available, [27] presents the impact of the network coding schemes on the field size and the delay. Packets can be combined dynamically using a sliding window mechanism, whereby the destination node acknowledges the degrees of freedom it receives and unnecessary packets are dropped from the sender queue, e.g., [28, 29, 30, 31]. For broadcasting data over a wireless network, [32] proposes a retransmission scheme with feedback that will guarantee the delivery of the data before a predefined deadline.

The network coding literature shows that robustness of the data trans-

mitted and the throughput supported by the network are significantly improved. However, the delay performance of the network coding protocols is not yet well understood because the previous results are typically (a) asymptotic in nature, meaning that the network parameters, such as the network size, the number of data packets to be transmitted, the field size, are unlimited or (b) focused mainly on the average delay performance. Nevertheless, the minimum and the maximum transmission delay are significant for non-asymptotic scenarios, especially for real-time applications with stringent delay constraints, e.g., live video streaming or distributed control in sensor-actuator networks.

The literature dedicated to network coding protocols used in data gathering applications is also limited. Data collection protocols have been deployed in [33, 34], where [33] presents a network coding protocol based on retransmissions and [34] provides an approach using network coding and duty-cycling in sensor networks. But, network coding protocols are suitable for broadcast applications. Thus, in multi-hop networks overhearing the transmissions by the nodes creates redundant packets, which increases the availability of the data and may help to complete faster the data transmission. In this sense, [35] offers a collection tree protocol using an algorithm based on overhearing, meaning that the intermediate nodes propagate not only its own data, but also data from their children. The protocol uses either a common RLNC or a systematic approach, where the nodes send first uncoded packets and then coded packets, over the field of size 16, i.e., $\mathbf{GF}(2^4)$. The results are implemented in wireless sensors using a grid network of 6 x 6 nodes and show the reliability of network coding for the data collection protocols. Moreover, [36] demonstrates the benefits of using network coding in terms of completion time in a large scale smart grid network, where the protocol employs Tunable Sparse Network Coding (TSNC) [37]. TSNC means sending sparse coded packets at the beginning of transmission and more dense to the end of the transmission. The level of sparsity is also exploited by using different feedback techniques. However, the settings of the tests are limited to $\mathbf{GF}(2)$, fixed number of packets used in combinations and feedback frequency, and just one data packet is initially available for transmission at each node.

Usually, data gathering applications need to face the challenges imposed by the entire system, e.g., large network size, limited data messages, specific coding parameters, different topologies, while minimizing the time to complete the data transmission. Since the previous results for data collection protocols using network coding are restricted, i.e., they use specific topology or particular parameters, a deeper understanding of the RLNC and the TSNC feasibility for the data gathering protocols is required for challenged networks.

Although reducing the active transmission time will provide energy savings, e.g., a lower energy per bit, these are hardly the only factors affecting the energy performance. This is particularly true in network coding protocols, which add a processing overhead for encoding, recoding, and decoding the data packets. Hence, previous literature has been focused on minimizing the energy consumption for the network coding protocols. For instance, [38] provides a theoretical and practical analysis in wireless ad-hoc networks under various settings, such as forwarding factor, managing data generations, and impact of transmission range. The authors characterizes the minimum amount of energy required to transmit a unit of data information in an one-to-many network. Furthermore, [39] proposes an algorithm on the oblivious back-pressure that reduces network power consumption over existing algorithms. In the context of body area networks, [40] demonstrates that for small networks, the amount of energy reduction achievable can range from 29% to 87% by using network coding techniques. Additionally, in the context of battery-powered networks without using network coding techniques, the energy is still a concern. For instance, significant literature in WSNs has been addressed the energy minimization using the radio characteristics, as shown in [41, 42, 43, 44]. Thus, [42] exploits techniques that turn off the radio during idle periods to reduce the power consumption, while [44] proposes a protocol that uses a radio wakeup mechanism to minimize the energy consumption. Moreover, [45] reduces the number of transmitted bits, [46] presents the real negligibility of the energy spent for switching transactions, and [47] shows the impact of the light and deep sleep modes for the MAC protocol in a WSN. Furthermore, data gathering protocols have been addressed in [48, 49, 50], where the routing schemes using the path less costly are provided for a low energy consumption in WSNs.

The related work on energy efficiency neglects several aspects. Communication engineers and researchers usually assume that the transmit power is the most important factor in determining the energy consumption in battery-powered networks, e.g, sensor networks. Therefore, in many cases, when designing protocols aiming to reduce the energy consumption of the overall network, the main approaches are focused on reducing the transmission of the data messages. Other sources of power consumption significant for the overall energy consumption of the system should be considered, such as the receive, listen, process power. For example, in WSNs, the receive power can be higher than the transmit power [51] and idle/listen power can almost equal to the receive power [52], but the list with examples can continue. Hence, the previous proposed models to characterize the total energy consumption of the system are not complete. This calls for a hardware abstraction that addresses the overall energy consumption of the network and allows for a

fast evaluation of the energy efficiency of the given protocol specifications. Moreover, a deeper analysis on how to perform network coding operations in battery-powered networks is missing. From the protocol perspective, the challenge is imposed by designing mechanisms to find a match between the protocol and the hardware for a low energy budget.

Network coding plays an important role for this thesis. On the one side, network coding can be used to improve the delay by minimizing the data transmission time or completing faster the collection of the data. The data transmission time analysis refers to intra-flow coding, while the completion time to inter-flow coding. On the other side, generating coded packets has an impact on the processing energy and consecutively, on the total energy consumption of the system. Thus, a deeper evaluation on how to implement coding operations is required. Protocol optimization needs both the delay implications and the energy impact. But, evidently, there is a trade-off between the delay and its implications on the energy consumption, because the longer the completion of transmission process takes, the more energy is consumed. Moreover, when using network coding, the impact of the field size on the transmission time and the energy consumption is also fundamental for this thesis. For instance, a higher field size implies a smaller delay, while a small field size a higher delay [53]. Additionally, for high field size, the trade-off between the transmission time and the energy is analysed in [54]. But, the influence of different field sizes on the energy efficiency is not known, e.g, the coding operations and the transmission time. Also, binary field size is simple to implement in practice because it only requires XOR operations, however, the impact of the field size on the energy consumption in practical scenarios needs to be better analysed.

1.3 Main Contributions and Thesis Outline

The focus of the thesis is on the design of the network coding protocols under different performance metrics and system constraints. The first two metrics refer to the delay. Thus, the first performance metric is the data transmission time of the network coding protocols, i.e., the time required to decode the data packets. This metric is not yet well understood because the previous results focused mainly on the average delay performance and are usually asymptotic in nature. But, we need to analyse the delay distribution, including the worst and best case scenarios. Thus, we call for the complete delay distribution using non-asymptotic scenarios, which is necessary for the applications with rigorous deadlines. Particularly, we propose to evaluate the delay behaviour of network coding in the case of limited finite

field sizes. Additionally, the second metric is the completion time, i.e., the total time required to collect all the data messages in a network. The evaluation of this metric using network coding protocols is unknown under different system constraints, e.g., coding parameters, topologies. We set out to understand the feasibility of two network coding protocols, RLNC and TSNC, for the data gathering applications and challenged networks. Including the presence of diverse feedback capabilities is useful for the implementation of these protocols in practical data gathering scenarios. Finally, the third metric is the energy consumption of the network coding protocols. The first two metrics support the third one because the energy consumption of the data transmission process is derived from the power consumption level and the active transmission periods of time. Besides, the network coding operations introduce additional processing costs. Hence, we call for further analysis the processing energy imposed by the network coding operations and the impact it has on the energy consumption of the overall system. These performance metrics aim to be evaluated in the presence of several constraints. In the following, we enumerate the constraints of the system, which refer to (i) the application (e.g., deadlines, topology, hardware limitation, network size, or length of the data messages); (ii) the network (e.g., data loss), and (iii) the protocol design (e.g., feedback type, field size, restrictions used in coding operations, and scheduling mechanisms). The ones for the application are usually fixed, defined by the designer according to the requirements of the system. As an example, for WSN applications, the hardware platforms are sensor motes and the size of the network can vary between few motes, e.g., body-area networks, to few hundreds or even more, e.g., forest sensing. Another example is smart grids, where the hardware of the nodes correspond to the smart meters or sensors and the size of the network is around hundreds or thousands of nodes. In case of the network constraints, these are challenges that cannot be controlled by the human being. On the contrary, the protocol limitations are the ones that can be tuned in the early phase of the communication protocol design. Not all the constraints are described for each of the metric previously defined, but during each chapter, they are properly explained. Thus, the main contributions of the thesis are presented in the following.

- *Analysis of the Delay Distribution for Broadcast Applications:* We investigate the delay distribution of RLNC for any field size and arbitrary number of encoded packets (or generation size). First, we take a combinatorial approach to the problem that allows us to derive the precise probability distribution of the delay of RLNC in the scenario of an erasure channel. We use this distribution to present some fundamental

properties of RLNC in erasure channels. Most notably, we show that for a small field size, the probability of having M linearly independent combinations after M received packets is already close to 1. Moreover, we demonstrate that to obtain a similar performance to the standard Automatic Repeat reQuest (ARQ) scheme, one can use RLNC in a finite field of small size, without the need of a feedback channel. Then, by introducing a Markov chain model we are able to obtain a complete solution for the erasure broadcast channel with two receivers. A comparison with ARQ with perfect feedback, round robin scheduling and a class of fountain codes reveals that network coding on $\mathbf{GF}(2^4)$ offers the best delay performance for two receivers. We also conclude that $\mathbf{GF}(2)$ induces a heavy tail in the delay distribution, which implies that network coding based on XOR operations although simple to implement bears a relevant cost in terms of worst-case delay. For the case of more receivers, which is mathematically challenging, we propose a brute-force methodology that gives the delay distribution of network coding for small generations and field size up to $\mathbf{GF}(2^4)$. The key idea for this method is to fix the pattern of packet erasures and to try out all possible encodings for various system and channel parameters. Our findings can be used to optimize network coding protocols with respect not only to their average but also to their worst-case performance.

- *Data Collection Protocol using Tunable Sparse Codes and Feedback Mechanisms:* After understanding the delay aspects of RLNC, we use the broadcast properties of network coding for data gathering applications. Here, we focus on TSNC which needs a deeper analysis for the evaluation of its applicability in practical scenarios. The idea is to model TSNC for various sparsity levels of the coded packets and different feedback types. We exploit two types of feedback, (1) the explicit feedback sent deliberately between nodes and (2) the implicit feedback when a node hears its neighbour transmissions. First, we propose analytical bounds for a line network, valid for any field size, various sparsity levels and the aforesaid feedback mechanisms. Second, we perform numerical results for a grid network using a small field size and compare the findings with some well-known protocols, such as RLNC and master-slave. We find that (a) the performance of TSNC depends on the topology of the network, (b) the completion time increases with the sparsity, and (c) the less frequent the explicit feedback, the higher the difference between sparse and dense coded packets. Hence, the minimum completion time is achieved either by using very sparse coded packets and frequent explicit feedback or dense coded packets

and a less frequent explicit feedback. Thus, our results show that it is possible to achieve a minimum completion time for a wide variety of sparsity-feedback pairs.

- *Hardware Abstraction Model for Protocol Design:* The design of energy-efficient protocols for data gathering and the development of hardware platforms for sensing applications are typically viewed as separate tasks. In many cases, communication engineers decide on the protocol specification based on high-level hardware abstractions, which assume that the radio transmission block is the top spender of electrical power. However, a closer inspection of the existing technology immediately reveals that this assumption does not hold for the majority of sensor platforms currently available in the market. To help close this gap, we propose a new hardware abstraction for protocol design that takes into consideration all of the main energy spending operations. Our results confirm that (a) the same protocol can have very different energy consumption profiles over different sensor platforms and (b) slight changes to the protocols using the proposed hardware abstraction can lead to significant energy gains over a wider range of sensor platforms. The latter findings are supported by real-life measurements, where network coding is implemented using common sensor motes, e.g., TelosB motes.

- *Protocol Optimization for Deadline Constrained Applications:* The design of the communication protocols in battery-powered networks usually neglects several key characteristics of the device's hardware. We show how to effectively reduce the total energy consumption of the network by designing energy-efficient protocols that use the previous hardware abstraction and mechanisms to optimize a communication protocol in terms of energy consumption. The problem is modelled for different feedback based techniques using a Markov Decision Process (MDP), where the nodes are directly connected to a base station, or through relays. The findings provide insights about the match between the protocol and the platform type. We show that the use of relays may decrease up to 4.5 times the total energy consumption by carefully matching the protocol and the hardware. The results are cross-validated using real-life measurements.

The remainder of the thesis is organized as follows. In Chapter 2, we analyse the delay distribution of network coding for broadcast applications using a closed-form expression, a Markov approach, and a brute-force model. Chapter 3 provides a data collection protocol for large scale applications focused on minimization of the completion time. In Chapter 4, we propose

a hardware abstraction model useful for the protocol design and with this model we optimize the protocol for deadline constraints applications in Chapter 5. Finally, Chapter 6 concludes the work and gives some directions for possible future work.

Parts of this work have been published in the following international journals:

- M. Nistor, D. E. Lucani, T. T. V. Vinhoza, R. A. Costa, and J. Barros, "On the Delay Distribution of Random Linear Network Coding", IEEE Journal on Selected Areas in Communications, 2011.
- M. Nistor, D. E. Lucani, and J. Barros, "Hardware Abstraction and Protocol Optimization for Coded Sensor Networks", IEEE/ACM Transactions on Networking, 2014.

Parts of this work have been published in the following international conferences:

- M. Nistor, J. Barros, F. Vieira, T. T. V. Vinhoza, and J. Widmer, "Network Coding Delay: A Brute-Force Analysis", Information Theory and Applications Workshop (ITA 2010), San Diego, CA, February 2010. Co-sponsored by IEEE.
- M. Nistor, R. A. Costa, T. T. V. Vinhoza, and J. Barros, "Non-Asymptotic Analysis of Network Coding Delay", IEEE Symposium on Network Coding, Toronto, June 2010.
- M. Nistor, D. E. Lucani, and J. Barros, "A Total Energy Approach to Protocol Design in Coded Wireless Sensor Networks". IEEE International Symposium on Network Coding, June, 2012.

Parts of this work are under submission:

- M. Nistor, D. E. Lucani, and J. Barros, "Network Coding Protocols for Data Gathering Applications", to be submitted to IEEE International Conference on Communications, 2015.

Other contributions:

- R. Prior, D. E. Lucani, Y. Phulpin, M. Nistor, and J. Barros, "Network Coding Protocols for Smart Grid Communications", IEEE Transactions on Smart Grids, 2013.

Chapter 2

Delay Distribution for Broadcast Applications

2.1 Motivation

After a decade of research, the throughput benefits and robustness properties of network coding [5, 21] have been well established for highly dynamic networks. In an information-theoretic setting, in which delay is not an issue, [5] showed that network coding is necessary and sufficient to achieve the multicast capacity of a general network. In general, the network coding delay is typically not a concern and, hence, this chapter focuses on a better understanding of this metric, which is critical in real-time applications.

Some previous delay implications are addressed in [22] and [23]. The proposed schemes use feedback techniques and aim to construct coded packets that are immediately decodable, meaning to provide a new information packet to the receiver. The online schemes provide a trade-off between the throughput achieved and the average delay experienced. For a half duplex channel, [24] shows the optimum number of coded packets, in terms of mean completion time, to be transmitted before the sender receives an acknowledgement. The work presented in [26] addresses a line network with different erasure probability channels and characterizes the average delay for a limited number of transmitted packets through the network. The findings show that the average delay is concentrated around its expectation. If the intermediate nodes in the network merely store and forward packets, fountain codes (e.g., Raptor codes [25]) offer an efficient alternative for end-to-end reliability, which relies on feedback solely for the purpose of terminating the encoding process.

This research effort has resulted in real-life protocols for wireless mesh

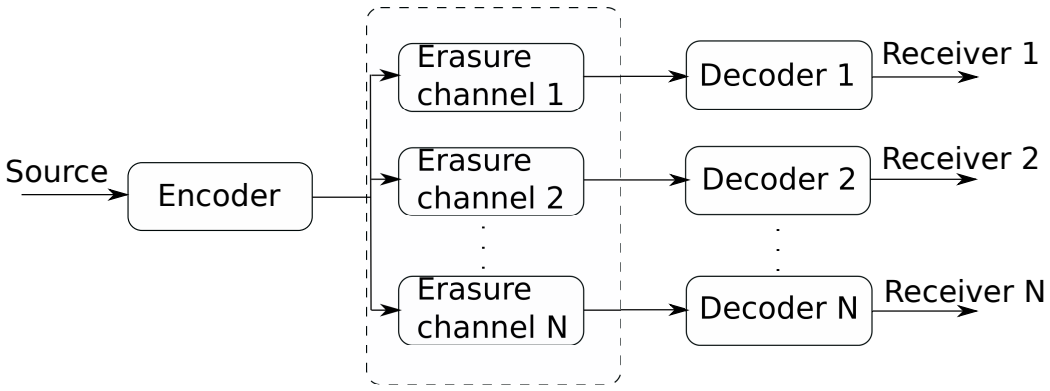


Figure 2.1: System model: one-to-many broadcast network.

networks [13] and peer-to-peer content distribution [55], among other applications [56]. Arguably less well understood is the delay behaviour of network coding, which is of particular relevance if network coding techniques are to be employed equally successfully in real-time applications such as live streaming or automatic control systems. The design of such systems requires knowledge not just of the average delay, which has already been studied to some extent, but also of the worst-case delay, which can be inferred from the complete delay distribution. Providing such a characterization of network coding delay for various scenarios of interest is the goal of this chapter.

Consider the broadcast scenario depicted in Figure 2.1. A source wants to transmit M packets $p_1, p_2, .., p_M$ to a set of receivers N. Each receiver observes the output of an independent erasure channel. To overcome the impairments of the channels while serving all of the receivers simultaneously, the source is allowed to mix the incoming packets and sends out linear combinations following the basic rules of RLNC. More specifically, the encoder mixes $p_1, p_2, .., p_M$ and outputs coded packets of the form $\sum_{i=1}^{M} \alpha_i \cdot p_i$. The coding coefficients $\alpha_1, \ldots, \alpha_M$ are independently and randomly selected from $\mathbf{GF}(q)$, where $q = 2^m$. The source appends in each coded packet the vector of coefficients, such that the receiver is able to reconstruct the linear combinations it has received.

Coded packets are transmitted over independent erasure channels. A packet is erased with probability e_j on channel j, $\forall j = 1, 2 \ldots N$. After collecting M linearly independent combinations, each decoder is able to reconstruct the encoding matrix by means of Gaussian elimination, thus recovering the original packets. Feedback is limited to one acknowledgement for the reception of all M packets. Our main figure of merit is the data transmission time, or simple the delay/decoding delay, D_j of each receiver j, which is defined as the total number of time slots required for j to decode

the M packets.

The remainder of the chapter is organized as follows. The related work is presented in the following. Section 2.3 provides the analysis of delay distribution for a scenario with a single receiver. The analysis is based on a closed-form expression and is compared against the ARQ scheme. The analysis of the delay distribution for a general case is given in Section 2.4. Here, the analysis is provided using a Markov Model and a Brute-Force Model. For the Markov Model, the delay performance of RLNC is compared against three reference schemes, namely, ARQ with perfect feedback, round robin scheduling, and a Luby Transform (LT) code. The chapter concludes with Section 2.5.

2.2 Related Work

Most results that address network coding delay take into account only the average delay performance. In [57] this metric is computed for a broadcast scenario with multiple receivers and then compared to round robin scheduling. The average delay of network coding is shown to decrease with a rising number of receivers. Likewise, the work described in [58] provides results for the average delay yet includes also the energy and throughput performance, as well as a comparison with standard ARQ schemes. The average delay for a time-division duplexing scheme is provided in [53] in a broadcast network and the case of one receiver as a function of the field size of RLNC is characterized without addressing the actual delay distribution. The contribution in [14] is focused on the average delay performance of systematic network coding with small field sizes, once again in a broadcast network.

When feedback is available, more sophisticated mechanisms can be used to broadcast a data stream to multiple receivers. A typical approach is for the receiver to send an acknowledgement once it decodes a complete set of coded packets (or generation). The work presented in [26] considers a limited number of packets and characterizes the average delay on a line network. It follows that in this special case the delay performance is concentrated around its expectation. Fountain codes, such as LT Codes [59], form yet another class of codes that provide reliable communication and throughput efficiency by acknowledging the successful decoding of the original message block.

Naturally, it is possible to explore feedback in a more elaborate way. For a half duplex channel, [24] combines the idea of incremental redundancy with network coding, i.e., using feedback to request additional coded packets. The work therein proves that there exists an optimal number of coded packets that can be transmitted before the sender receives an acknowledge-

ment. There, the optimum is defined in terms of the average delay required to complete the transmission of a generation of packets. What each receiver feeds back is the number of the degrees of freedom that are still required for decoding the entire generation successfully. An extension to this work is presented in [58], which offers a complete delay and energy characterization of the aforementioned coding scheme. Online network coding mechanisms for random arrivals of packets are considered in [10, 29, 30, 31]. These contributions assume that packets are combined dynamically using a sliding window mechanism, whereby the destination node acknowledges the degrees of freedom it receives. Unnecessary packets are thus dropped from the sender queue. The delay implications of this online network coding mechanism are addressed in [22] and [23].

Since most results in the literature are based on the average delay, the worst case delay performance is still not well understood. Previous work on worst case delay includes [60] which uses deterministic network calculus to describe delay service guarantees for a packet at an intermediate node, however the network model therein does not admit packet losses. By considering erasure channels our work offers results for a more realistic network model. This scenario of a source broadcasting packets to several receivers over erasure channels was analyzed in [61], where it was shown that the minimization of delay for this broadcast scenario is NP-hard. Thus, knowing the complete delay distribution is clearly a step-forward, as it allows us to give upper and lower delay bounds for such a model.

2.3 Closed-Form Expression for a Single Receiver Case

The main contributions for this part of the work are mentioned in the following:

- *Delay Distribution:* We use a combinatorial approach to derive the probability distribution for the delay of RLNC, in the scenario of an erasure channel. The results are valid for any number of packets and any finite field size.

- *Field size:* We prove that a modest field size is sufficient for the probability of achieving optimum delay to be very close to 1.

- *Binary field has a heavy tail:* We show that the binary field, which is attractive due to the low complexity of the XOR based encoding procedure, induces a delay distribution with a heavy tail, which makes it

hard to provide quality of service guarantees, with respect to decoding delay.

- *Comparison with ARQ:* We show that delay distribution of RLNC for a small finite field size, $\mathbf{GF}(2^4)$ is close to that of ARQ schemes with perfect feedback.

2.3.1 Problem Statement and Main Result

Throughout this work, we consider the scenario of a source that has to transmit M packets to a receiver over an erasure channel. RLNC is used as a coding strategy for recovering from erasures in the channel, when feedback is not available (except for the acknowledgement the reception of all the M packets). In each time slot, the source transmit a packet of the form $s = \sum_{i=1}^{M} \alpha_i \cdot p_i$. We assume that there is at least one i for which $\alpha_i \neq 0$. If after the random selection process, $\forall i,\ \alpha_i = 0$, the source restarts the selection process until $\exists i : \alpha_i \neq 0$.

Our performance metric, the delay, is denoted it by D. Notice that the receiver needs to obtain M linearly independent combinations of the initial packets to be able to decode, this implies that $D \geq M$. Our goal is to derive the probability distribution of the delay, $\mathcal{P}(D = k)$, for $k \geq M$.

We present some quantities that will be instrumental for the description of our main result. Let $S_k(t)$ denote the set of all possible numbers of linearly independent coded packets received by time slot t such that the delay is k. We will prove in Section 2.3.2 that $S_k(t)$ is given by

$$S_k(t) = \begin{cases} \{0, \ldots, t\} & \text{if } t \leq k - M \\ \{M-(k-t), \ldots, t\} & \text{if } k-M<t\leq M-1 \\ \{M-(k-t), \ldots, M-1\} & \text{if } M-1<t<k \\ \{M\} & \text{if } t = k. \end{cases} \tag{2.1}$$

We define $f_t(n)$ as the number of coded packets that are linearly independent with respect to any subset of received coded packets that provide n degrees of freedom at time slot t (where n is such that the delay is k), and $g_t(n)$, as the number of linearly dependent coded packets. We will prove in Section 2.3.2 that $f_t(n)$ and $g_t(n)$ are given by

$$f_t(n) = \begin{cases} q^M - 1 - g_t(n) & \text{if } n \in S_k(t), \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

and

$$g_t(n) = \begin{cases} \sum_{j=1}^{n} \binom{n}{j} (q-1)^j & \text{if } n \in S_k(t) \text{ and } n \geq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

We are now ready to present our main result.

Theorem 1. *Consider a link with capacity of one packet per time slot. A packet is dropped with probability e. For a source that has M packets to transmit and uses RLNC (in the finite field $\mathbf{GF}(q)$), we have that the probability distribution of the delay, D, for $k \geq M$ is given by*

$$\mathcal{P}(D=k) = \sum_{j=0}^{k-M} \binom{k-1}{j} \frac{q^M - q^{M-1}}{(q^M-1)^{k-j-1}} C_1(1) \cdot e^j (1-e)^{k-j}, \tag{2.4}$$

where $C_1(1)$ can be computed from the following recursion:

$$\begin{cases} C_{k-1}(M-1) &= 1 \\ C_{k-1}(s) &= 0, \text{ for } s < M-1 \\ C_t(M-1) &= \prod_{j=t}^{k-2} g_j(M-1), \text{ for } t \leq k-2 \\ C_t(s) &= f_t(s) C_{t+1}(s+1) + g_t(s) C_{t+1}(s), \\ & \quad \text{for } t \leq k-2 \text{ and } s < M-1. \end{cases} \tag{2.5}$$

The implications of this result are discussed in Section 2.3.3.

2.3.2 Proof of Theorem 1

The proof is based on a series of lemmas.

We separate the channel effect (due to the erasures) from the possibility of receiving a coded packet that is linearly dependent with the ones previously received.

Lemma 1. *Let E represent the number of erasures in the first $k-1$ time slots. We have that*

$$\mathcal{P}(D=k) = \sum_{j=0}^{k-M} \mathcal{P}(D=k-j | E=0) \cdot \binom{k-1}{j} e^j (1-e)^{k-j}.$$

Proof. Let $E(k)$ represent the "Erasure occurs in time slot k". It follows

that

$$\mathcal{P}(D{=}k) = \mathcal{P}(D{=}k|E(k))\mathcal{P}(E(k)) + \mathcal{P}(D{=}k|\overline{E(k)})\mathcal{P}(\overline{E(k)}).$$

Notice that, if an erasure occurs in time slot k, the receiver gets no new coded packet in that time slot and, therefore one of the two following statements is true: (a) if the receiver has already decoded all the packets at time slot k, then it has decoded all the packets before time slot k; or (b) the receiver has not decoded all the packets at time slot k. In case (a) is true, we have that the delay is smaller than k, and in case (b) is true, we have that the delay is larger than k. Therefore, we have that $\mathcal{P}(D = k|E(k)) = 0$ and thus

$$\mathcal{P}(D = k) = \mathcal{P}(D = k|\overline{E(k)})\mathcal{P}(\overline{E(k)}) = \mathcal{P}(D = k|\overline{E(k)})(1 - e). \quad (2.6)$$

Let E represent the number of erasures occurred in the first $k - 1$ time slots. We have that

$$\mathcal{P}(D{=}k|\overline{E(k)}) = \sum_{j=0}^{k-1} \mathcal{P}(D{=}k|E = j, \overline{E(k)})\mathcal{P}(E = j|\overline{E(k)}).$$

First, notice that $\mathcal{P}(E = j|\overline{E(k)}) = \mathcal{P}(E = j)$ since erasures in different time slots are independent. Moreover, we have that E is binomial distributed and hence $\mathcal{P}(E = j) = \binom{k-1}{j} e^j (1 - e)^{k-1-j}$. Now, notice that if $E > k - 1 - (M - 1) = k - M$, in the first $k - 1$ slots, the receiver gets less than $M - 1$ coded packets, which implies that the delay is larger than k. Therefore, we have that, for the delay to be k, we must have $E \leq k - M$, which implies that

$$\mathcal{P}(D = k|\overline{E(k)}) = \sum_{j=0}^{k-M} \mathcal{P}(D = k|E = j, \overline{E(k)}) \cdot \binom{k-1}{j} e^j (1 - e)^{k-1-j}.$$

Now, we focus on $\mathcal{P}(D = k|E = j, \overline{E(k)})$. The linear combinations transmitted in the time slots where the j erasures occurred do not play any role in this probability, since the receiver did not get them. Therefore, having a delay equal to k in the presence of j erasures, is equivalent to having a delay of $k - j$ in the special case of a perfect channel, where no erasure occurs. Hence, we may conclude that $\mathcal{P}(D = k|E = j, \overline{E(k)}) = \mathcal{P}(D = k - j|E = 0)$

and thus

$$\mathcal{P}(D=k|\overline{E(k)})=\sum_{j=0}^{k-M}\mathcal{P}(D=k-j|E=0)\cdot\binom{k-1}{j}e^j(1-e)^{k-1-j}. \quad (2.7)$$

To conclude the proof, we just need to replace (2.7) in (2.6) and the result follows. □

With the previous result, we are able to study the probability distribution of delay through the analysis of the perfect channel case, i.e. we only need to characterize $\mathcal{P}(D=k|E=0)$.

We start by describing what are the possible values for the degrees of freedom at the receiver, at a given time slot and for a fixed delay value.

Lemma 2. *Let $S_k(t)$ represent the set of all possible numbers of linearly independent combinations of packets received by time slot t so that the delay is k (in a channel without erasures). Then we have that $S_k(t)$ is given by* (2.1).

Proof. To ensure that the delay is equal to k, we must have that, at the end of time slot $t=k$, the receiver has obtained M degrees of freedom so that he can decode all the packets. Thus, $S_k(k)=\{M\}$.

Now, consider the case $t\leq k-M$. In this case, there are still at least $k-(k-M)=M$ time slots. Therefore, even if in all the first t slots there was an erasure, the receiver may still be able to obtain delay k. Therefore, we have that for $t\leq k-M$, $S_k(t)=\{0,\ldots,t\}$.

For $t>k-M$, the number of remaining time slots (to obtain delay k) is given by $k-t$, which is less than M. Therefore, to obtain a delay equal to k, we must have that the number of independent linear combinations at time slot t is at least $M-(k-t)$. Moreover, to ensure that the delay is k, the receiver can only obtain the last degree of freedom at time slot k. Therefore, we have that, for $k-M<t\leq M-1$, $S_k(t)=\{M-(k-t),\ldots,t\}$, and for $M-1<t\leq M-1$, $S_k(t)=\{M-(k-t),\ldots,M-1\}$. To conclude the proof, just notice that the expressions derived here for $S_k(t)$ are precisely the ones presented in (2.1). □

Next, we describe the total number of possible coded packets that can be innovative to the receiver, given that it has already a certain number of degrees of freedom.

Lemma 3. *Consider the case where n linearly independent combinations of packets have been received by time slot t. Let $f_t(n)$ represent the number of coded packets that are linearly independent from any subset of the received*

coded packets, such that the delay is k, and $g_t(n)$ the number of linearly dependent ones. Then, we have that

$$f_t(n) = \begin{cases} q^M - 1 - g_t(n) & \text{if } n \in S_k(t), \\ 0 & \text{otherwise.} \end{cases}$$

and

$$g_t(n) = \begin{cases} \sum_{j=1}^{n} \binom{n}{j} (q-1)^j & \text{if } n \in S_k(t) \text{ and } n \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. First, notice that, by *Lemma 2*, if $n \notin S_k(t)$ then it is impossible to obtain delay k. Therefore, we have that if $n \notin S_k(t)$, then $f_t(n) = g_t(n) = 0$.

There are $q^M - 1$ possible linear combinations of the source packets (we exclude the combination with all coefficients null). Therefore, it is clear that $f_t(n) = q^M - 1 - g_t(n)$ (if $n \in S_k(t)$). Thus, our focus must be given to the number of different linear combinations that can be constructed with the so far received packets, given that the receiver has already n degrees of freedom. Let $\mathcal{S}$ be the subspace of $\mathbf{GF}(q)^M$ that represents the current knowledge of the receiver and let $s_1, \ldots, s_n$ be elements of the basis of $\mathbf{GF}(q)^M$. Any linear combination of the received coded packets can be written as $\sum_{j=1}^{n} \alpha_j \cdot s_j$, where the operations are taken in $\mathbf{GF}(q)$ and $\exists j : \alpha_j \neq 0$. We refer to the size of the set $\{\alpha_j \neq 0\}$ as the *degree* of the linear combination. Consider a certain value for the degree, say j. Each of the j non-null coefficients may take $(q-1)$ different values. Moreover, there are $\binom{n}{j}$ different possible choices for which coefficients are non-null. Therefore, we have that for a degree j, there are $\binom{n}{j}(q-1)^j$ different linear combinations that can be constructed from the coded packets received thus far and, thus, the result follows. □

We are now ready to present the expression for $\mathcal{P}(D{=}\,k|E{=}\,0)$.

Lemma 4. *We have that $\mathcal{P}(D = k|E = 0) = \frac{q^M - q^{M-1}}{(q^M-1)^{k-1}} \cdot C_1(1)$, where $C_1(1)$ can be computed from the recursion in* (2.5).

Proof. Let $C(k, M)$ be the number of possible sets of k linear combinations such that the delay is equal to k. We have that $\mathcal{P}(D = k|E = 0) = \frac{C(k,M)}{(q^M-1)^k}$. Let $C_t(s)$ be the number of possible sets of linear combinations to be transmitted in time slots $t+1, \ldots, k-1$ such that in time slot $k-1$ the receiver has $M-1$ linearly independent coded packets, given that in time slot t the receiver has s linearly independent coded packets. Notice that, in the first time slot, the received linear combination is always innovative

and there are $q^M - 1$ possibilities. Moreover, to have delay equal to k, in time slot $k-1$ the receiver must have $M-1$ degrees of freedom and the received linear combination in time slot k must be linearly independent from the previous ones, and by *Lemma 3*, there are $q^M - q^{M-1}$ possibilities. Therefore, we have that $C(k, M) = (q^M - 1) \cdot (q^M - q^{M-1}) \cdot C_1(1)$ and thus $\mathcal{P}(D = k|E = 0) = \frac{q^M - q^{M-1}}{(q^M-1)^{k-1}} \cdot C_1(1)$.

Now, we need to show that $C_1(1)$ can be computed by (2.5). By the definition of $C_t(s)$, we have that $C_{k-1}(M-1) = 1$ and $C_{k-1}(s) = 0$, for $s < M-1$. Regarding $C_t(M-1)$ for $t \leq k-2$, we have that, if in time slot t we have $M-1$ linearly independent combinations, then we must receive linearly dependent combinations up to slot $k-1$ to have a delay equal to k. Therefore, by *Lemma 3*, we have that $C_t(M-1) = \prod_{j=t}^{k-2} g_j(M-1)$. Notice that, for certain values of s and t, it may not be possible to obtain delay equal to k, but this is already taken into account since $g_t(M-1) = 0$ if $M-1 \notin S_k(t)$, and by *Lemma 2*, $S_k(t)$ is the possible values for the degrees of freedom at the receiver at time slot t, such that we can obtain delay equal to k.

Let us now focus on $C_t(s)$ for $t \leq k-2$ and $s < M-1$. If in time slot t, the receiver has s degrees of freedom in its buffer, then at time slot $t+1$ it has s or $s+1$ degrees of freedom. The former case implies that in time slot t a linearly dependent combination was transmitted (by *Lemma 3*, there are $g_t(s)$ possible combinations for that). The later case implies that the transmitted linear combination is linearly independent, and there are $f_t(s)$ possibilities. Therefore, we may conclude that $C_t(s) = f_t(s) \cdot C_{t+1}(s+1) + g_t(s) \cdot C_{t+1}(s)$.

Once again, for certain values of s and t, it may not be possible to obtain delay equal to k, but this is already taken into account since $f_t(s) = 0$ or $g_t(s) = 0$ if $s \notin S_k(t)$. To conclude the proof, just notice that the recursion described here is precisely the one described in (2.5). □

With this set of results we have proven *Theorem 1*.

2.3.3 Numerical Results

Given a set of parameters (e.g., the number of packets to transmit over a network, the erasure probability of the channel, and the field size), we are interested in knowing the probability distribution of delay. To clarify the relevance of our work we demonstrate how we can use the results derived in previous sections to improve the design of systems with delay requirements. By analyzing the delay of RLNC schemes we are able to find encoding parameters that comply with some required quality of service (QoS). Possible

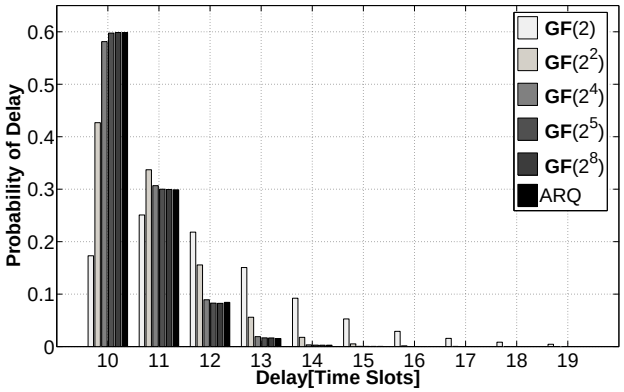
applications include distributed control systems, where delay bounds must be ensured (e.g. a master node needs to guarantee that his commands reach the slave nodes before a deadline) and real-time multimedia streaming services with stringent delay constraints. It is remarkable that the obtained expression of the delay from the previous section can be used to build a bound of the decoding delay for the case of one receiver, where the extension to the multiple-receiver case still remains a big challenge.

The rest of this section is devoted to the analysis of the delay distribution under the variation of different parameters (field size, erasure probability, and number of packets). The obtained results are compared with the ARQ scheme, known to achieve optimal throughput and minimum delay over erasure channels if and only if perfect feedback is available, where by perfect feedback we mean a lossless and instantaneous feedback.

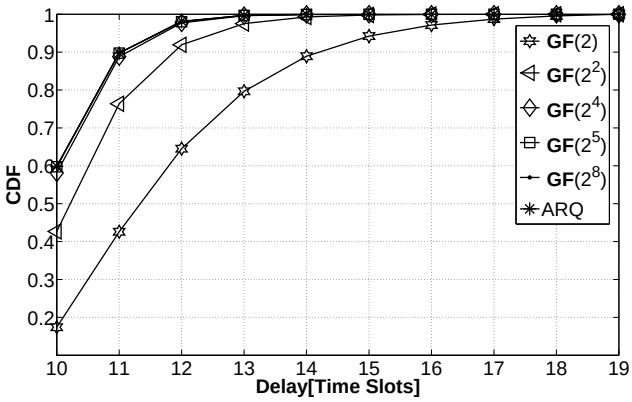
2.3.3.1 The Effect of Field Size

In order to study the effect of field size in delay when using RLNC, we will consider a fixed number of packets, $M = 10$, a fixed erasure probability $e = 0.05$, and $e = 0.2$ and let the field size vary. Figure 2.2 plots the probability of delay when we consider an erasure probability of $e = 0.05$. From the plot we can observe that for field size $\mathbf{GF}(2)$, we obtain non-negligible delay with a heavy tail. For instance, the probability of having delay $D = 10$ is around 17% and of having delay $D = 14$ is almost 9%. Moreover, for $\mathbf{GF}(2^4)$ and $D = 10$ the probability is 56%, whereas for $D = 14$ it is already close to zero. With ARQ we have a probability distribution of delay for $D = 10$ around 60%, whereas for $D = 14$ we get almost 0%. The cumulative distribution function from Figure 2.2(b) shows that for fields larger than $\mathbf{GF}(2^4)$, the probability of receiving all the packets within $D = 14$ time slots is almost 99%, where for $\mathbf{GF}(2)$, the probability is around 90%. Both histogram and cumulative function reveal that for $q \geq 16$ the probability distribution functions become very close to that of ARQ. A similar discussion follows for higher erasure probability, as shown by Figure 2.3. We find the probability of $D = 10$ for $\mathbf{GF}(2)$ achieves almost 3.1%, whereas for fields larger than $\mathbf{GF}(2^4)$ the value lies between 10% and 10.7%. Moreover, for $D = 19$ and $\mathbf{GF}(2)$ we get 5% and for $\mathbf{GF}(q \geq 2^4)$ the probability is close to 1%. For the same parameters, the probability for ARQ schemes is 10.9% and 1%, respectively. As the probability of erasure increases, we can see that it becomes the dominant term in (2.4).

From our illustrations we can see that for a field size $\mathbf{GF}(q \geq 2^4)$ the gain of RLNC in terms of delay becomes negligible. The size of the field $\mathbf{GF}(q)$ does have implications from the point of view of computational complexity.



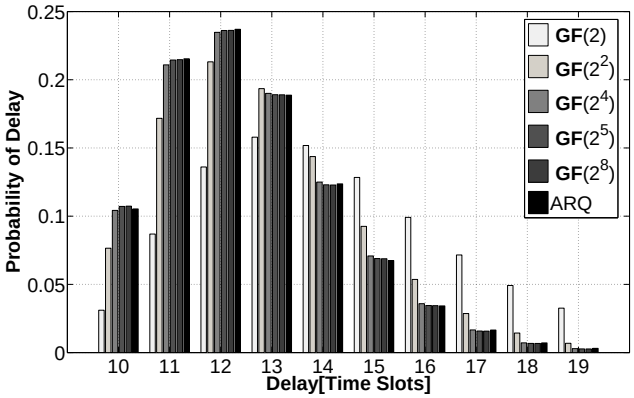
(a) Histogram of probability distribution of delay



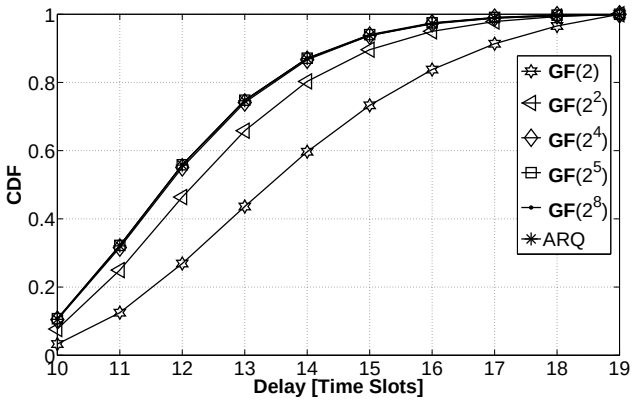
(b) CDF of probability distribution of delay

Figure 2.2: Analysis for 10 packets, 0.05 erasure probability, and various fields size.

The difference in implementation and system complexity between $\mathbf{GF}(2^4)$ and $\mathbf{GF}(2^8)$ is not very significant, because both cases require Gaussian elimination. Another issue of course is the overhead incurred by placing all the coefficients in the header of the coded packet. This obviously grows with the field size. Both Figure 2.2 and Figure 2.3 show that the gap between $\mathbf{GF}(2)$ and $\mathbf{GF}(2^8)$ becomes smaller as the erasure probability increases. It is clear that RLNC for field size larger than $\mathbf{GF}(2^4)$ approaches ARQ schemes with perfect feedback. Small field sizes, such as $\mathbf{GF}(2)$ are very easy to implement due to the simple XOR encoding operations. However, the analysis shows a delay distribution with heavy tail, which means that delay guarantees cannot be easily provided.



(a) Histogram of probability distribution of delay

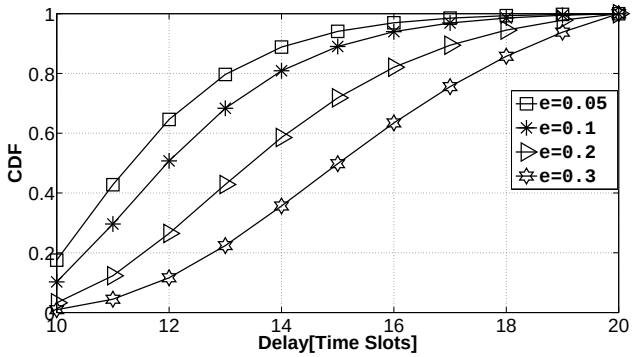


(b) CDF of probability distribution of delay

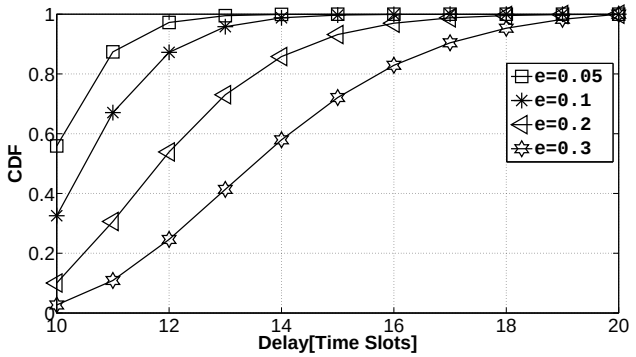
Figure 2.3: Analysis for 10 packets, 0.2 erasure probability, and various fields size.

2.3.3.2 The Effect of Erasure Probability

The illustrations from Figure 2.4 show RLNC for fixed number of packets, $M = 10$, fixed field size, and varying erasure probability ($e = 0.05$, $e = 0.1$, $e = 0.2$, and $e = 0.3$). We present only $\mathbf{GF}(2)$ and $\mathbf{GF}(2^4)$, because it is clear $\mathbf{GF}(2^4)$ is representative for higher field sizes. The result shows that the effect of erasures and of the field sizes are completely separable. In this case, by increasing the erasure probability, the average delay increases with the same proportion for $\mathbf{GF}(2)$ as for $\mathbf{GF}(2^4)$. For instance for the case of $\mathbf{GF}(2)$ the average delay for $e = 0.05$ is 12.20, whereas for $e = 0.3$ we get 16.39. An increase by 35% is observed. The results for the case of $\mathbf{GF}(2^4)$ are similar, where the average delay varies between 10.60 and 14.38, resulting again in an increase of 35%.



(a) CDF of probability distribution of delay for $\mathbf{GF}(2)$

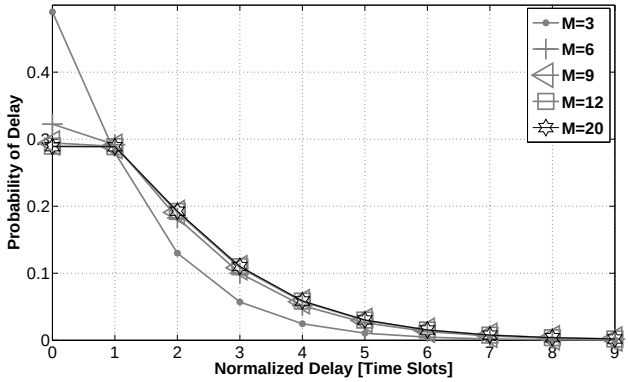


(b) CDF of probability distribution of delay for $\mathbf{GF}(2^4)$

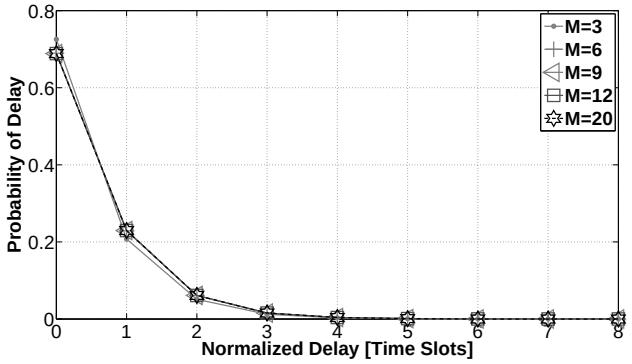
Figure 2.4: Analysis for 10 packets, fixed field size, and different erasure probability.

2.3.3.3 The Effect of the Number of Packets

The effect of different number of packets over different field sizes is presented in this subsection. For a perfect channel (without erasures) we plot the probability distribution of normalized delay for receiving M linearly independent combinations of packets, as shown in Figure 2.5. By normalized delay we mean the number of coded packets subtracted from the delay value $(D-M)$. The results for $\mathbf{GF(2)}$ show asymptotic behavior only at $M = 9$, whereas for $\mathbf{GF(2^2)}$ the asymptotic convergence starts already at $M = 3$. The higher the value of the field, the lower the value of M for which the probability becomes constant. The probability of having M linearly independent combinations after M received coded packets is already close to 1. For field size larger than $\mathbf{GF}(2^2)$, obviously the value is almost constant for all M packets.



(a) PDF of Normalized Delay for $\mathbf{GF}(2)$



(b) PDF of Normalized Delay for $\mathbf{GF}(2^2)$

Figure 2.5: Normalized delay analysis for perfect channel.

2.3.4 Discussions

The next natural step would be to extend the delay analysis to the case of $N > 2$ receivers. Although the erasure channels are independent, they all share a common source. Therefore, the outputs of the channels are not independent. Suppose that Y_j is the output of the channel observed by j. We have, for instance, that $\mathcal{P}(Y_1 = 0, Y_2 = 1) = 0$, because we consider an erasure channel (no bit flipping), but $\mathcal{P}(Y_1 = 0) > 0$ and $\mathcal{P}(Y_2 = 1) > 0$. Hence, $\mathcal{P}(Y_1 = 0, Y_2 = 1) \neq \mathcal{P}(Y_1 = 0) \cdot \mathcal{P}(Y_2 = 1)$.

Moreover, although the erasure patterns observed in each channel are independent, the fact that a linear combination is innovative for one receiver may imply that it is also innovative for another receiver. We illustrate this fact through the following example. Consider the case of two receivers and say that the source has transmitted p_1 and $p_2 + p_3$ in the first two time

slots. Assume that receiver 1 has observed no erasure and that receiver 2 has received only $p_2 + p_3$. Let $A_i(t)$ represent the event "the linear combination transmitted in time slot t is innovative for receiver i". Since the knowledge space of receiver 2 is a subspace of the one of receiver 1, we have that $A_2(3) \subset A_1(3)$. Thus, these events are not independent. The same applies to the delays of different receivers, because the incurred delay depends essentially on whether or not the received linear combinations are innovative. Although for large field sizes all randomly generated linear combinations are innovative with probability close to one, this is not the case for small fields such as $\mathbf{GF}(2)$, which are practically relevant due to the low encoding and decoding complexity.

We conclude that a trivial extension of the main arguments in the proof for the single-receiver case does not yield a correct solution for the multiple user case. Solving the multiple receiver case requires taking complex statistical dependencies into consideration, which at this time seems elusive.

2.4 General Receiver Case

We start with an analytical analysis provided by a Markov Model and then we show the results using a Brute-Force Model.

2.4.1 Markov Model

Seeking to characterize the probability distribution of the delay of RLNC for the aforementioned communications scenario, we make the following contributions:

- *Fundamental Analysis:* We propose a Markov chain approach that enables us to derive the aforementioned delay distribution for the case of two receivers and independent erasure channels. The one receiver case follows immediately as a special case.

- *Performance Evaluation and Comparison:* In the case of two receivers, we demonstrate that RLNC outperforms ARQ with perfect feedback for a Galois field larger or equal to $\mathbf{GF}(2^2)$. The performance of RLNC is also shown to be superior to that of LT codes (a class of fountain codes [59]) and round robin scheduling — irrespective of the field size. Our results also show that opting for network coding over $\mathbf{GF}(2)$, which is convenient for its low computational complexity, bears the cost of a heavy tail in the delay distribution.

In this section we show that the delay distribution problem can be cast in a Markov chain model. This approach allows us to obtain the delay distribution for RLNC. More specifically, we provide an exact characterization for two receivers over independent erasure channels. The one-receiver instance is obtained as a special case. The following definitions are useful.

Definition 1. The *knowledge space* (or simply the *knowledge*) K_l of a receiver l at a given time t is defined as the linear span of the linear combinations of packets $p_1, p_2, .., p_M$ received by l until time t.

Definition 2. We say that a node has k *degrees of freedom (dofs)* if the dimension of its knowledge space is k.

Definition 3. We say that a linear combination is *innovative* to receiver l at time t if it does not belong to K_l.

We also require the following events related to the arrival of a new coded packet or linear combination.

Definition 4. Let E_K denote the event that occurs when a received linear combination is innovative given the knowledge space K_l of receiver l.

Definition 5. Let E_K^{nz} denote the event that occurs when a received linear combination, with a coding vector that is not all zeros, is *not* innovative with respect to the knowledge space K_l of receiver l.

Definition 6. Let Z denote the event that corresponds to the arrival of a linear combination with an all-zeros coding vector.

Since in our broadcast setting the source is common to all receivers, it is likely that subsets of receivers have the same information at any given time. We formalize this intuition as follows.

Definition 7. We say that a subset of receivers $L \subseteq R_l, L \neq \emptyset$ and $|L| > 1$ share the *common knowledge* C_L at a given time t if $C_L = \cap_{\{l \in L\}} K_l$ at time t.

2.4.1.1 N-Receiver Case

We consider the scenario depicted in Figure 2.1, in which a source wants to transmit M packets to N receivers. The transmission adds a degree of freedom (dof) to the knowledge space of a receiver if the channel does not erase it and the sent linear combination is linearly independent of all previously received linear combinations. We can describe this process by means of a Markov chain model.

A Markov chain model is defined by a set of states and a set of transitions with given probabilities. In our case, a state consists of the numbers of dofs at each receiver, the number of dofs for the common knowledge space of pairs of receivers, the number of dofs for the common knowledge space of groups of 3 receivers, and so forth until we reach the number of dofs for the common knowledge space of all receivers. The dependence between the receivers inherent to the broadcast scenario is captured by these state variables of common knowledge.

Each state is described by a set of elements. By $(i_1, i_2, \ldots, i_N)$ we represent the dofs for each of the N receivers, with $i_l = \dim(K_l)$, where K_l is the knowledge of receiver l, denoted here as R_l. We write

$$c_{1,2} = \dim(C_{1,2}) \ \ldots \ c_{1,2,\ldots,N} = \dim(C_{1,2,\ldots,N})$$

for the common knowledge between each combination of $2, 3, \ldots, N$ receivers. The total number of indexes of a state is given by the expression $\sum_{\gamma=0}^{N-1} \binom{N}{\gamma} = 2^N - 1$, which results in a total number of states of $(M+1)^{2^N-1}$. The first state of the model represents 0 dof for all receivers, naturally the common dofs are also 0, hence the state can be showed as $(0, 0, ..., 0)$.

A transition to other states depends on the previous state, on the set of receivers for which the coded packet was correctly received and the subset of nodes that obtain an innovative linear combination. In every time slot a transition occurs. The maximum dofs at each receiver is reached when M linearly independent information packets are received. When all nodes receive M dofs, they all share the same knowledge. Thus, there exists an absorbing state, which is $(M, M, ..., M)$. A state β of a Markov chain is called absorbing if it has a transition probability $p_{\beta,\beta} = 1$, which implies that the process never changes state once it reaches state β.

Transition Probability Matrix

As the state space is finite, we can represent the transition probability distribution by the transition matrix $\mathbf{T}$, whose (u, v) element gives the probability of going from state u to state v. Since Markov chain is stationary, the transition matrix $\mathbf{T}$ does not change with time.

From the $(M+1)^{2^N-1}$ states we discard those never entered by the process. We call these states invalid states and focus on the valid ones. We denote by A the number of valid states. Notice that $A \leq (M+1)^{2^N-1}$. For the lower bound, the number of valid states should be greater than $(M+1)^N$, which corresponds to the total number of N-tuples representing the individual knowledges of the receivers for M transmitted packets. Hence, A is

bounded by two exponential (on N) growth terms:

$$(M+1)^N \leq A \leq (M+1)^{2^N-1}. \tag{2.8}$$

Lemma 5. *For the $N = 2$ receivers case, the number of valid states is the solution of a difference equation. For $M \geq 2$, $A(M, N)$ is given by:*

$$A(M,2) = 10 + \frac{47}{6}(M-2) + 2(M-2)^2 + \frac{1}{6}(M-2)^3. \tag{2.9}$$

Proof. The number of valid states is the solution of a difference equation when $N = 2$ and $M \geq 2$, and we denote it by $A(M, 2)$. By considering the number of valid states for $M = 2, 3, 4, 5, 6$ packets, we noticed that they follow a difference equation of the form $A(M+5,2) = A(M+4,2) - 4A(M+3,2) + 6A(M+2,2) - 4A(M+1,2) + A(M,2)$. The characteristic polynomial for this difference equation is $z^4 - 4z^3 + 6z^2 - 4z + 1 = 0$, which has only one root, $z = 1$, of multiplicity 4. Therefore, the linear recurrence equation with constant coefficients takes the form

$$A(\theta, 2) = c_1 + c_2\theta + c_3\theta^2 + c_4\theta^3, \tag{2.10}$$

with $\theta \in 0...3$. From (2.10) we find $c_1 = 10$, $c_2 = \frac{47}{6}$, $c_3 = 2$ and $c_4 = \frac{1}{6}$ after applying the initial conditions, e.g., the number of valid states for $M = 2, 3, 4, 5, 6$. By substituting the coefficients and $\theta = M - 2$ in expression (2.10) we get $A(M,2) = 10 + \frac{47}{6}(M-2) + 2(M-2)^2 + \frac{1}{6}(M-2)^3$, which is the same with (2.9). This concludes the proof. □

The general expression for the transition probability matrix of size $A \times A$ is

$$\mathbf{T} = \begin{bmatrix} p_{1,1}(1) & p_{1,2}(1) & ... & p_{1,A}(1) \\ \vdots & ... & \vdots & \vdots \\ p_{A,1}(1) & p_{A,2}(1)... & p_{A,A-1}(1) & p_{A,A}(1) \end{bmatrix}, \tag{2.11}$$

where $p_{j,a}(1)$ denotes the probability of arriving at state a after one step when the chain starts in state j. Here, state 1 corresponds to $(0, 0, \ldots, 0)$ and state A corresponds to $(M, M, , \ldots, M)$. As state A is an absorbing state, we have that $p_{A,A}(1) = 1$ and $p_{A,a}(1) = 0$, $\forall a = 1, 2, ..., A-1$.

A k-step transition probability matrix can be computed as $\mathbf{T}^k$, i.e., the k-th power of the transition matrix. This k-step transition probability matrix represents the probability of arriving at each of the states in k transitions

(time slots). The expression for matrix $\mathbf{T}^k$ is

$$\mathbf{T}^k = \begin{bmatrix} p_{1,1}(k) & p_{1,2}(k) & ... & p_{1,A}(k) \\ \vdots & ... & \vdots & \vdots \\ 0 & ... & 0 & 1 \end{bmatrix}, \tag{2.12}$$

where $p_{j,a}(k)$ denotes the probability of reaching state a in k steps after starting from state j. We are particularly interested in the case of $p_{1,A}(k)$ because it describes the probability that the packets are successfully decoded after k time slots.

Delay Distribution

Our goal is to derive a probability distribution for the delay, i.e., determining $\mathcal{P}(D = k)$ for M packets, where $k \geq M$ represents the number of time slots needed to decode the information. Let us formalize our definition of delay and its link to the Markov chain model.

A delay D of k time slots indicates that exactly k time slots are required for all receivers to decode the information, i.e., to transit to state $(M, ..., M)$ of the Markov chain for the first time. The probability of $\mathcal{P}(D \leq k) = p_{1,A}(k)$, which is the probability of arriving in state $(M, ..., M)$ of the Markov chain given that the system started at state $(0, ..., 0)$. Thus, the probability of decoding in exactly k time slots is given by $\mathcal{P}(D = k) = \mathcal{P}(D \leq k) - \mathcal{P}(D \leq k - 1)$.

Computational Complexity

For large N, the model requires high computational complexity, because the number of valid states A increases exponentially with the number of receivers. The impact is evident when computing the transition probability matrix and managing the operations with matrices. We want to find $\mathcal{P}(D \leq k)$, which requires us to multiply a $A \times A$ matrix up to $(k - 1)$ times. For instance, $N = 3$ requires already $O(M^7)$ states, which means a state has 7 elements, 3 for the knowledge of receivers and 4 for the common knowledge. This is still feasible for the two-receiver case, as we discuss in Section 2.4.1.2.

Particular Cases

Two special cases are observed when the channels assign specific values for the erasure probability and a third one emerges when the receivers are arranged in a different way:

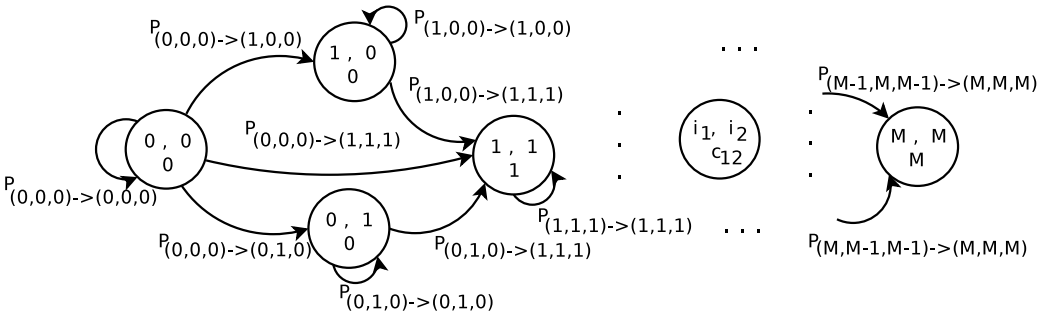


Figure 2.6: Markov Chain for two-receiver case.

- Perfect channels: It is the case when no erasures occur. A state is represented only by the knowledge of one receiver and the Markov chain has $(M+1)$ states.

- Noisy channels: This situation corresponds to channels with high erasure probabilities, whose delay can be approximated by independent random variables. The cumulative distribution function of the N random variables is then the product of the cumulative distribution function of delay for each one of the N receivers. Hence, the Markov chain reduces to the case of one receiver and has $(M+1)$ states.

- Degraded channels: This case refers to the situation where the channel of receiver l is a degraded version of the channel of $l-1$, $\forall l = 2, 3, ...N$. In this case, the maximum delay is always the delay of receiver N. The Markov chain reduces again to the case of only one receiver and has $(M+1)$ states.

In these particular situations, we can observe that our Markov chain model is represented by one receiver. The total number of states equals $M+1$. For that reason, the model can be easily extended to accommodate an arbitrary number of receivers.

2.4.1.2 Two-Receiver Case

For this case we denote by K_1 the knowledge of the first receiver, by K_2 the knowledge of the second receiver and by $C = K_1 \cap K_2$ the common knowledge of both receivers. In this case, each state is described by 3 elements (i_1, i_2, c), with $i_1 = \dim(K_1)$, $i_2 = \dim(K_2)$ and $c = \dim(C)$. The first state corresponds to $(0, 0, 0)$. The elements (i_1, i_2, c) evolve in each slot and the final state is defined by (M, M, M).

The following theorem states the possible transition probabilities for the two-receiver case. Let d_1 and d_2 denote the dimensions of the *non* common knowledge of R_1 and R_2, respectively. This means that $d_a = \dim(K_a \setminus C)$ and $i_a = c + d_a$, where $i_a = \dim(K_a)$, $c = \dim(C)$ and $a \in \{1, 2\}$.

Theorem 2. *In the Markov model RLNC over* $\mathbf{GF}(q)$ *with two receivers, there exist at most* 7 *states to which state* (i_1, i_2, c) *can transit to with non-zero probability. The transition probabilities are given by (2.13),*

$$\mathcal{P}_{(i_1,i_2,c)\to(i_1',i_2',c')} = \tag{2.13}$$

$$\begin{cases} e_1e_2 + e_1(1-e_2)\mathcal{P}(E^{nz}_{K_2} \cup Z) + e_2(1-e_1)\mathcal{P}(E^{nz}_{K_1} \cup Z) + \\ \quad +(1-e_1)(1-e_2)\mathcal{P}(E^{nz}_{K_1} \cap E^{nz}_{K_2} \cup Z), & \text{if (1)} \\ (1-e_1)(1-e_2)\mathcal{P}(E_{K_1} \cap E_{K_2} \cap E_{K_1 \cup K_2}), & \text{if (2)} \\ (1-e_1)(1-e_2)\mathcal{P}(E_{K_2} \cap E^{nz}_{K_1}) + e_1(1-e_2)\mathcal{P}(E_{K_1} \cap E_{K_2} \cap E^{nz}_{K_1 \cup K_2}), & \text{if (3)} \\ e_1(1-e_2)\mathcal{P}(E_{K_2} \cap E_{K_1 \cup K_2}), & \text{if (4)} \\ (1-e_2)(1-e_1)\mathcal{P}(E_{K_1} \cap E^{nz}_{K_2}) + e_2(1-e_1)\mathcal{P}(E_{K_1} \cap E_{K_2} \cap E^{nz}_{K_1 \cup K_2}), & \text{if (5)} \\ e_2(1-e_1)\mathcal{P}(E_{K_1} \cap E_{K_1 \cup K_2}), & \text{if (6)} \\ (1-e_1)(1-e_2)\mathcal{P}(E_{K_1} \cap E_{K_2} \cap E^{nz}_{K_1 \cup K_2}), & \text{if (7)} \end{cases}$$

where

- *(1)* $i_1' = i_1, i_2' = i_2, c' = c$,
- *(2)* $i_1' = i_1 + 1, i_2' = i_2 + 1, c' = c + 1$,
- *(3)* $i_1' = i_1, i_2' = i_2 + 1, c' = c + 1$,
- *(4)* $i_1' = i_1, i_2' = i_2 + 1, c' = c$,
- *(5)* $i_1' = i_1 + 1, i_2' = i_2, c' = c + 1$,
- *(6)* $i_1' = i_1 + 1, i_2' = i_2, c' = c$,
- *(7)* $i_1' = i_1 + 1, i_2' = i_2 + 1, c' = c + 2$.

and

- $\mathcal{P}(E^{nz}_{K_a} \cup Z) = (q^{-M+c+d_a})$,
- $\mathcal{P}(E_{K_a} \cap E_{K_a \cup K_b}) = 1 - q^{-M+c+d_a+d_b}$,
- $\mathcal{P}(E_{K_a} \cap E_{K_b} \cap E_{K_a \cup K_b}) = 1 - q^{-M+c+d_a+d_b}$,

- $\mathcal{P}(E_{K_a} \cap E^{nz}_{K_b}) = \frac{(1-q^{-M+c+d_a})(q^{-M+c+d_b}-q^{-M+c})}{(1-q^{-M+c})}$,
- $\mathcal{P}(E^{nz}_{K_a} \cap E^{nz}_{K_b} \cup Z) =$
  $= (q^{-M+c+d_b}) - \left[\frac{(1-q^{-M+c+d_a})(q^{-M+c+d_b}-q^{-M+c})}{1-q^{-M+c}}\right]$,
- $\mathcal{P}(E_{K_a} \cap E_{K_b} \cap E^{nz}_{K_a \cup\ K_b}) =$
  $= \left[\frac{(1-q^{-M+c+d_a})(1-q^{-M+c+d_b})}{(1-q^{-M+c})} - (1- q^{-M+c+d_a+d_b})\right]$,
  with $a, b \in \{1, 2\}$ *and* $a \neq b$.

Proof. The first part of the proof is combinatorial in nature and relies on considering all possible events, namely (a) independent channels suffering erasures, and (b) the coded packet adding a dof, (c) the coded packet not adding a dof, or (d) a coding vector of all zeros with respect to the vector spaces $K_1, K_2, K_1 \cup K_2$. We also observe the fact that several combinations generate the same transition and that at every time slot the source can provide at most one dof to each receiver.

Let $f = \dim(K_1 \cup K_2) \leq M$. Note that $f = i_1 + i_2 - c = d_1 + d_2 + c$ or equivalently, $c = i_1 + i_2 - f$. Let us also define $f' = \dim(K_1 \cup K_2 \cup v)$, where v is the incoming coded packet. Note that the transitions to i'_a based on i_a are straight-forward, i.e. either $i'_a = i_a$, because of an erasure or the fact that no additional dof is provided to K_a, or $i'_a = i_a + 1$, if there is no erasure and the coded packet is innovative to K_a. When an incoming coded packet v is innovative to $K_1 \cup K_2$ this implies that $f' = f + 1$. Conversely, if it does not provide a dof or the coding vector is all-zero then $f' = f$. Using this knowledge, we can determine the values of c' based on the transition to i'_1, i'_2 and f'. Thus, there are 3 possible values for c', namely, $c, c + 1, c + 2$. If both receivers maintain the same dofs after the transition, clearly $c' = c$ because $f' = f$, $i'_1 = i_1$ and $i'_2 = i_2$. When an incoming coded packet adds a dof to $K_1 \cup K_2$, i.e., $f' = f + 1$, then we have two possibilities: (i) $c' = c$ corresponding to the case in which only one receiver, say a, gets a new dof, because $i'_a = i_a + 1$, $i'_b = i_b$, and $b \neq a$, or (ii) $c' = c + 1$ in case both receivers get a new dof, because $i'_1 = i_1 + 1$ and $i'_2 = i_2 + 1$.
When the incoming coded packet is not innovative to $K_1 \cup K_2$, i.e., $f' = f$, then (i) $c' = c+1$ if only one receiver, say a, gets a new dof because $i'_a = i_a+1$, $i'_b = i_b$, and $b \neq a$ or (ii) $c' = c + 2$ if both receivers get a new dof, because $i'_1 = i_1 + 1$ and $i'_2 = i_2 + 1$.

Thus, a state has at most 7 transition states with non-zero transition probability, including the case of self-transition. The transition probabilities match the previously described events, combining the effect of erasures and innovativeness of the coded packets.
This concludes the first part of the proof. □

Let us now prove the expressions for the probabilities of the different events for a coded packet in terms of the knowledge at the receivers. The probability that a coded packet is innovative with respect to the knowledge space K_a of receiver a is given by

$$\mathcal{P}(E_{K_a}) = \mathcal{P}(E_{K_a} \cap E_C) = 1 - q^{-M+c+d_a}. \tag{2.14}$$

The event that a coded packet is not innovative or that the coding vector is all-zero is the negation of an innovative coded packet arriving at the receiver. Thus, we get

$$\mathcal{P}(E_{K_a}^{nz} \cup Z) = q^{-M+c+d_a}. \tag{2.15}$$

The probability of a coded packet not adding a dof to one receiver's knowledge space while it is already a part of the other receiver's knowledge space is given by

$$\begin{aligned}
\mathcal{P}(E_{K_a} \cap E_{K_b}^{nz}) &= \mathcal{P}(E_{K_a} \cap E_{K_b}^{nz} \cap E_C) \\
&= \mathcal{P}(E_{K_a}|E_{K_b}^{nz} \cap E_C)\mathcal{P}(E_{K_b}^{nz}|E_C)\mathcal{P}(E_C) \\
&= \mathcal{P}(E_{K_a}|E_C)\mathcal{P}(E_C)\mathcal{P}(E_{K_b}^{nz}|E_C) \\
&= \frac{1-q^{-M+c+d_a}}{1-q^{-M+c}}(1-q^{-M+c})\mathcal{P}(E_{K_b}^{nz}|E_C).
\end{aligned}$$

Given that $\mathcal{P}(E_{K_b}^{nz}|E_C) = 1 - \mathcal{P}(E_{K_b}|E_C) = 1 - \frac{1-q^{-M+c+d_b}}{1-q^{-M+c}}$, then

$$\mathcal{P}(E_{K_a} \cap E_{K_b}^{nz}) = \frac{(1-q^{-M+c+d_a})(q^{-M+c+d_b} - q^{-M+c})}{(1-q^{-M+c})}. \tag{2.16}$$

The probability that a coded packet is not innovative for both receivers or that the coding vector is all-zero is given by the expression

$$\begin{aligned}
&\mathcal{P}(E_{K_a}^{nz} \cap E_{K_b}^{nz} \cup Z) = 1 - [\mathcal{P}(E_{K_a} \cap E_{K_b}) + \mathcal{P}(E_{K_a} \cap E_{K_b}^{nz}) + \\
&+ \mathcal{P}(E_{K_b} \cap E_{K_a}^{nz})] = \mathcal{P}(E_{K_b}^{nz}) - \mathcal{P}(E_{K_b}^{nz} \cap E_{K_a}).
\end{aligned}$$

Taking into consideration (2.15) and (2.16), the final result is given by

$$\begin{aligned}
&\mathcal{P}(E_{K_a}^{nz} \cap E_{K_b}^{nz} \cup Z) = \\
&= \left(q^{-M+c+d_b}\right) - \left[\frac{(1-q^{-M+c+d_a})(q^{-M+c+d_b} - q^{-M+c})}{1-q^{-M+c}}\right].
\end{aligned} \tag{2.17}$$

The probability that a coded packet is innovative for both receivers while adding a dof to $K_a \cup K_b$ is given by the expression $\mathcal{P}(E_{K_a} \cap E_{K_b} \cap E_{K_a \cup K_b}) = 1 - q^{-M+c+d_a+d_b}$.
For the case of a coded packet that adds a dof to the knowledge of both receivers but does not add a dof to $K_a \cup K_b$ we have

$$\begin{aligned} & \mathcal{P}(E_{K_a} \cap E_{K_b} \cap E^{nz}_{K_a \cup K_b}) = \\ = \ & \mathcal{P}(E_{K_a} \cap E_{K_b} \cap E_C) - \mathcal{P}(E_{K_a} \cap E_{K_b} \cap E_{K_a \cup K_b}) \\ = \ & \mathcal{P}(E_{K_a}|E_{K_b} \cap E_C)\mathcal{P}(E_{K_b}|E_C)\mathcal{P}(E_C) - P(E_{K_a} \cap E_{K_b} \cap E_{K_a \cup K_b}) \\ = \ & \mathcal{P}(E_{K_a}|E_C)P(E_{K_b} \cap E_C) - \mathcal{P}(E_{K_a} \cap E_{K_b} \cap E_{K_a \cup K_b}). \end{aligned}$$

The probability of a coded packet being innovative to K_a, given that is also innovative to the common knowledge of both receivers is

$$\mathcal{P}(E_{K_a}|E_C) = \frac{\mathcal{P}(E_{K_a} \cap E_C)}{P(E_C)} = \frac{1 - q^{-M+c+d_a}}{1 - q^{-M+c}}. \tag{2.18}$$

The probability $\mathcal{P}(E_{K_a} \cap E_{K_b} \cap E^{nz}_{K_a \cup K_b})$ is obtained by combining (2.14) and (2.18):

$$\begin{aligned} & \mathcal{P}(E_{K_a} \cap E_{K_b} \cap E^{nz}_{K_a \cup K_b}) = \\ = \ & \left[\frac{(1 - q^{-M+c+d_a})(1 - q^{-M+c+d_b})}{(1 - q^{-M+c})} - (1 - q^{-M+c+d_a+d_b}) \right]. \end{aligned} \tag{2.19}$$

The probability that a coded packet is innovative with respect both to K_a and to $K_a \cup K_b$ is equivalent to the probability that is innovative with respect to K_a, K_b and $K_a \cup K_b$ yielding

$$\begin{aligned} & \mathcal{P}(E_{K_a} \cap E_{K_a \cup K_b}) = \\ = \ & \mathcal{P}(E_{K_a} \cap E_{K_b} \cap E_{K_a \cup K_b}) = (1 - q^{-M+c+d_a+d_b}). \end{aligned} \tag{2.20}$$

This concludes the proof. Thus, we can compute the probability distribution of the delay (2.12). But first let us provide some intuition on the transition probabilities for each state (i_1, i_2, c) to state (i'_1, i'_2, c') given by the following 7 cases:

1. $i'_1 = i_1$, $i'_2 = i_2$, $c' = c$: this case includes the events that (i) both channels induce erasures, (ii) one channel induces erasure and the receiver corresponding to the other channel gets a coded packet that does not increase the dofs of its knowledge or was encoded with all zero coefficients, or (iii) the coded packet is not innovative for both receivers or was encoded with all zero coefficients.

2. $i'_1 = i_1 + 1$, $i'_2 = i_2 + 1$, $c' = c + 1$: both receivers get innovative coded packets and the transmitted coded packet is innovative with respect to $K_1 \cup K_2$.

3. $i'_1 = i_1$, $i'_2 = i_2 + 1$, $c' = c + 1$: this case considers the events that (i) both receivers get the coded packet and it is innovative for R_2 but not to R_1, (ii) only R_2 receives the coded packet and it is innovative for R_2 but does not add a dof to K_1, or (iii) only R_2 receives the coded packet and the coded packet adds a dof to both K_1 and K_2 but is not innovative for $K_1 \cup K_2$.

4. $i'_1 = i_1$, $i'_2 = i_2+1$, $c' = c$: R_2 gets a new coded packet that is innovative for K_1, K_2 and $K_1 \cup K_2$, and the channel associated to R_1 suffers an erasure.

5. $i'_1 = i_1+1$, $i'_2 = i_2$, $c' = c+1$: this case is symmetric to 3).

6. $i'_1 = i_1 + 1$, $i'_2 = i_2$, $c' = c$: this case is symmetric to 4).

7. $i'_1 = i_1 + 1$, $i'_2 = i_2 + 1$, $c' = c + 2$: both R_1 and R_2 receive the coded packet. The coded packet is innovative to both receivers but is not innovative for $K_1 \cup K_2$.

Note that in case 7) the common part increases by 2 in a single time slot. This happens if the knowledge of both receivers is increased, i.e., the coded packet is innovative to both receivers, but it is already part of the common knowledge space. For instance, when $M = 2$ we can have the following situation. Before receiving a new coded packet, R_1 has $K_1 = \{p_1\}$ and R_2 has $K_2 = \{p_2\}$, therefore $K_1 \cap K_2 = \emptyset$. This is represented by state $(1, 1, 0)$, i.e., each receiver has one dof and they share no common knowledge. In the next time slot, a new coded packet is transmitted that is a linear combination of p_1 and p_2. The knowledge of both receivers increases by one once this coded packet is received, while the common part is increased by 2 since $\dim(K_1 \cap K_2) = \dim(p_1, p_2) = 2$. This state is represented by $(2, 2, 2)$.

2.4.1.3 One-Receiver Case

Suppose now that the source wants to send M packets to a receiver. A similar model Markov model, focusing on average delay, is studied in [53]. The transitions from i_1 to state i'_1 are given by the following two cases:

- $i'_1 = i_1$: the coded packet suffers an erasure or the coded packet is received but it does not add a new dof to the knowledge of the receiver.

- $i_1' = i_1 + 1$: a coded packet is received and the receiver gets a new dof.

The associated transition probabilities are given by

$$\mathcal{P}_{(i_1)\to(i_1')} = \begin{cases} e_1 + (1-e_1)\mathcal{P}(E_{K_1}^{nz} \cup Z), & \text{if } i_1' = i_1 \\ (1-e_1)\mathcal{P}(E_{K_1} \cup Z), & \text{if } i_1' = i_1 + 1 \end{cases}$$

$$= \begin{cases} e_1 + (1-e_1)q^{-M+i_1}, & \text{if } i_1' = i_1 \\ (1-e_1)(1-q^{-M+i_1}), & \text{if } i_1' = i_1 + 1. \end{cases}$$

As in the general case, the probability $\mathcal{P}(D \leq k)$ is given by the element $p_{1,A}(k)$ from $\mathbf{T^k}$, where $A = M + 1$. The $(M+1)$-th state is associated with state $i_1 = M$ and the first state is associated to state $i_1 = 0$.

2.4.1.4 Insights and Practical Implications

Having derived analytical expressions for the delay distribution of RLNC, we are now ready to discuss some of the insights they offer. To this end, we compare the delay of RLNC with three well established transmission schemes:

- **ARQ**: The sender will transmit every packet, whose reception is not acknowledged by each receiver. This scheme is known to achieve optimal throughput and minimum delay over the erasure channel with one receiver (see [30]).

- **LT codes**: This class of rateless codes is well known to ensure reliability and optimum throughput over erasure channels (see [25]). Feedback is only used for acknowledging the successful reception of all packets. The encoder performs combinations of packets using an appropriate degree distribution in order to minimize the number of redundant code packets. Usually, the degree is chosen from a robust soliton distribution, which requires two additional parameters to be set, namely a constant, $const < 1$, and a bound δ on the decoding failure probability [62].

- **Round robin**: This mechanism is well known to yield optimum throughput in broadcast scenarios ([57], [63]), where the receivers experience the same probability of erasure. It is a scheduling scheme with minimalistic feedback, where the source sends the packets in round-robin fashion until the receivers announce the reception of all M packets.

The numerical results for these three techniques were obtained through simulation. For the ARQ scheme, the source sends the same packet until it

receives ACKs of successful reception from all the receivers. In the case of the LT codes we use the robust soliton distribution, with $const = 0.01$ and $\delta = 0.7$. As our analysis focuses on small values of M, the performance of LT codes is not sensitive to the values of *const* and δ. The round robin experiences the same minimalistic feedback as our model by only acknowledging the correctly received packets at all the receivers. Using these reference systems as a benchmark for the delay performance of RLNC, we now discuss the two-receiver case in detail.

Two Receivers

We characterize the delay distribution of RLNC by means of the transition probability matrix defined in (2.12). The analysis is carried out for different field sizes and erasure probabilities using the cumulative distribution function (CDF).

Field Size: In order to study the effect of the field size, we assume that the channels have identical erasure probability. The number of transmitted packets is fixed to 10 and we vary the field size from $\mathbf{GF}(2)$ to $\mathbf{GF}(2^8)$. A comparison with ARQ, round robin scheduling and LT codes is also included. For the case of a channel with small erasure probability (equal to 0.05), the findings are shown in Figure 2.7(a), where it is possible to see that RLNC in $\mathbf{GF}(2^4)$ outperforms ARQ, and RLNC over $\mathbf{GF}(2)$ outperforms LT codes. For a scenario with higher erasure probability, Figure 2.7(b) shows that RLNC outperforms ARQ when done in fields as small as $\mathbf{GF}(2^2)$. LT codes and the scheduling scheme are outperformed by RLNC.

For field sizes larger than $\mathbf{GF}(2^4)$, the delay performance is similar, as evidenced by both figures. Thus, in the following we take $\mathbf{GF}(2^4)$ as a representative for higher fields. We conclude from here that for $\mathbf{GF}(2^4)$ and a broadcast scenario with two receivers, RLNC outperforms all the other transmission schemes. In particular, $\mathbf{GF}(2)$ performs better than LT codes and round robin. Moreover, as the erasure probability increases, the minimum field size under which RLNC outperforms ARQ becomes smaller.

Erasure Probability: The effect of erasure probability is illustrated in Figure 2.7(c). We fix the number of packets ($M = 10$) and the field size ($\mathbf{GF}(2)$ and $\mathbf{GF}(2^4)$), and vary the erasure probability ($e = 0.05$, $e = 0.1$, $e = 0.2$ and $e = 0.3$). The channels have identical erasure probabilities. We present only $\mathbf{GF}(2)$ and $\mathbf{GF}(2^4)$, because for $\mathbf{GF}(q \geq 2^4)$ the results are similar to $\mathbf{GF}(2^4)$. In this case, by increasing the erasure probability, the average delay increases with the same proportion for $\mathbf{GF}(2)$ and for $\mathbf{GF}(2^4)$. For instance, if we consider the case of $\mathbf{GF}(2)$, the average delay for $e = 0.05$ is 12.68 and for $e = 0.3$ we get 18.40, which implies an increase of 45%. The

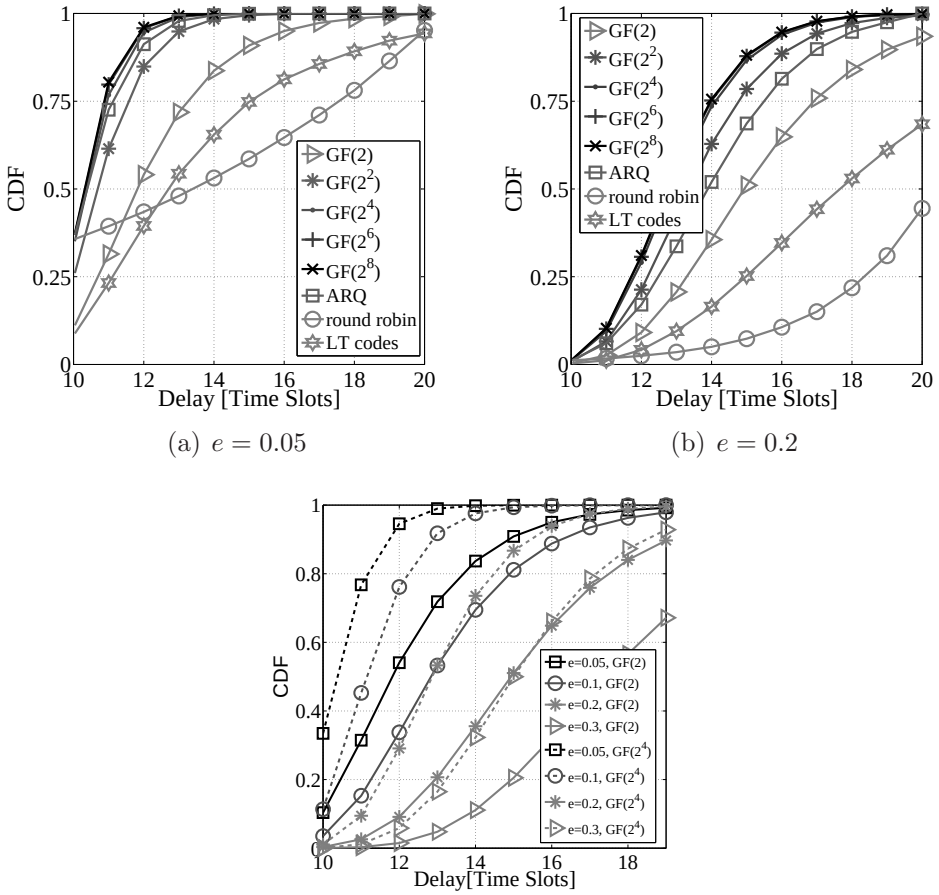


Figure 2.7: (a) and (b) The effect of the field size in the two-receiver case, for $M = 10$ packets, various field sizes, and erasure probability (a) $e = 0.05$, (b) $e = 0.2$. (c) The effect of erasure probability in the two-receiver case with $M = 10$ packets, fixed field sizes and various erasure probabilities.

same analysis for $\mathbf{GF}(2^4)$ yields 10.96 and 15.77, respectively corresponding to an increase of 44%. Hence, for two receivers, the result shows that the effect of erasures and field size are completely separable.

One Receiver

In order to study the effect of field size in delay when using RLNC, we fix the number of packets, $M = 10$, and erasure probability, $e = 0.2$, while varying the field size. From our illustrations in Figure 2.8 we can see that $\mathbf{GF}(2)$ induces a heavy tail. The average delay for $\mathbf{GF}(2)$ is 14.31 time-slots and for

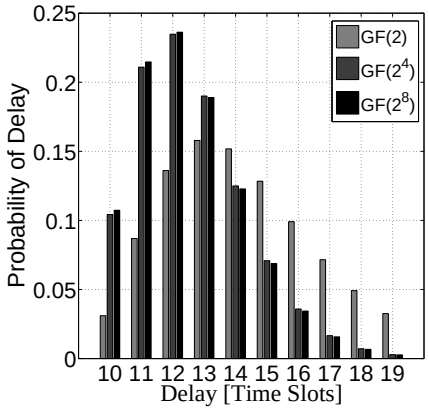


Figure 2.8: The effect of the field size in the one-receiver case with $M = 10$ packets, erasure probability $e = 0.2$.

$\mathbf{GF}(2^4)$ and $\mathbf{GF}(2^8)$ is 13.5 time-slots.

Number of Packets

We now analyze the effect of the number of packets to be encoded for various field sizes. The probability that a receiver obtains M linearly independent combinations from M received coded packets is given by: $\mathcal{P} = \prod_{i=0}^{M-1}(1 - q^{i-M})$ [57], where q is the field size. By changing variables and calling $u = i - M$ we have: $\mathcal{P} = \prod_{u=-M}^{-1}(1 - q^u)$. Note that for a given field size q, if u is below a certain value, q^u is negligible when compared to 1. Hence, when we increase the number of packets, i.e. the value of M, the probability of having M linearly independent combinations from M coded packets remains the same. For $\mathbf{GF}(q \geq 2^2)$, we may safely neglect the effect of the number of packets in the probability of obtaining M linearly independent combinations from M coded packets. For the delay in noisy channels, the erasure probability plays the central role on the choice of a suitable M in order to ensure, for example, that all M packets are delivered on time when the application has strict deadlines.

2.4.2 Brute-Force Model

Given the mathematical difficulty of characterizing the distribution of the network coding delay for multiple receivers, we propose the following brute-force approach. We consider the following figures of merit. The *delay* D_j of receiver j, also called the decoding delay, is defined as at the beginning of this chapter. Particular attention shall be devoted to the maximum delay,

taken among all the receivers. The *in-order delivery delay*, denoted by $D_{p_i}(j)$, corresponds to the number of time slots necessary for receiver j to deliver p_i in the correct order to the application. The maximum in-order delivery delay for p_i is then given by $D_{p_i} = \max_j D_{p_i}(j)$. We focus on the multiple-receiver case and make the following contributions:

- *Brute-force analysis of delay:* Since the previous analysis for the single-receiver case does not carry over to the case of multiple receivers, we propose a brute-force approach, whereby the erasure pattern is fixed and the delay is measured for all possible encodings (i.e. sets of linear combinations of packets).
- *Curve Fitting:* For small field sizes, e.g., $\mathbf{GF}(2)$ and limited number of receivers, the delay distribution is shown to be well approximated by a normal distribution.
- *Engineering Implications:* We discuss how knowing the distribution of delay can improve the design of network coding based protocols for real-time applications.

2.4.2.1 Methodology

We start by fixing the number of receivers N, the field size, the number of packets to be combined, and the erasure probability of each channel. Based on the values of these parameters, we generate erasure patterns for testing purposes, whose length t_s must be sufficient to allow for all decoders to recover all sent packets in one generation. Each erasure pattern can be represented as a matrix

$$\underline{E} = [\underline{\epsilon}_1|\underline{\epsilon}_2|\dots|\underline{\epsilon}_N]^T, \tag{2.21}$$

where $\underline{\epsilon}_i$ is a binary vector of length t_s representing for each time slot whether receiver i obtained an erasure or a correct packet.

In a second step, we generate all possible encodings, i.e., sets of linear combinations that can be generated by the encoder. The maximum block length of an encoding is limited to the number of packets we want to transmit.

Finally, we carry out Gaussian elimination for all receivers, erasure patterns and encodings. The delay is measured by counting the number of slots until each packet is decoded and taking the maximum thereof. To reduce the computational efforts, we adapt the number of time-slots to the parameter values of each experiment, such that all the packets are received with more than 99% probability. The minimum number of time-slots required is computed using Algorithm 1.

Algorithm 1 Number of time slots for an experiment using Brute-Force Model

Data: erasure probability (e), number of packets (M), number of receivers (N)
Result: number of time-slots (t_s)
$Sum \leftarrow 0$
$TotalSum \leftarrow 0$
$t_s \leftarrow M - 1$
while $TotalSum \leq 0.99$ **do**
  $t_s \leftarrow t_s + 1$
  $Sum \leftarrow Sum + \binom{t_s-1}{M-1} \cdot (1-e)^M \cdot e^{t_s-M}$
  $TotalSum \leftarrow Sum^N$
end while

2.4.2.2 Experiments

The proposed brute-force approach is obviously computationally heavy and is only feasible for small field size and limited number of receivers. The number of packets to be encoded, also known as generation size, must also be kept to a small value. Nevertheless, as we shall demonstrate in the next section, the approach provides new insights into the performance of random network coding.

Our experiments focus on the cases of 2, 3 or 4 receivers, generations of 2 or 3 packets and erasure probabilities of 0.05, 0.1, and 0.2. Each experiment is denoted as aNbM, where a and b give the number of receivers and the number of packets, respectively.

For each erasure probability we run a number of experiments (one for each erasure pattern) and compute the mean and standard deviation of the delay. The length of the erasure pattern is determined using Algorithm 1. To ensure the statistical significance of our results, we must run a sufficient number of experiments depending on the value of the erasure probability. Let K be the number of erasures in a specific erasure pattern. We start by setting $K = 0$ and compute the probability of occurrence for this erasure pattern. The procedure is repeated by incrementing K at each step until the total probability of the thus obtained erasure pattern is higher than 90%. This can be validated using the binomial distribution to compute the total probability of the erasure patterns with up to K erasures, specifically

$$\mathcal{P}_o = \sum_{i=0}^{K} \binom{N \cdot ts}{N \cdot ts - i} \cdot (1-e)^{N \cdot ts - i} \cdot e^i. \tag{2.22}$$

The value of K for which this probability exceeds 90% is denoted as K_{max}. It follows that the number of erasure patterns to be used for experimentation is given by

$$E = \sum_{i=0}^{K_{max}} \binom{N \cdot t_s}{N \cdot t_s - i}.$$

Table 2.1: Cases of interest for Brute-Force Model: the number of erasure patterns, probability of the erasure pattern $\mathcal{P}_o(e)$ and the probability that decoding fails $\mathcal{P}(\overline{D})$

Experiment and e		Number of erasure patterns (E)	$\mathcal{P}_o(e)$ [%]	$\mathcal{P}(\overline{D})$ [%]
3N2M	0.05	79	98	4.5
	0.1	576	94.4	4.18
	0.2	31.180	94.8	6.09
4N2M	0.05	137	95.7	5.5
	0.1	6.196	95.6	5.8
	0.2	536.155	91.1	7.5
2N3M	0.05	11	91.3	4.9
3N3M	0.05	121	96.3	9.8

The cases of interest and the corresponding experimentation values are listed in Table 2.1.

2.4.2.3 Evaluation

The delay is measured in each experiment for every encoding and every receiver. The delay histogram is then obtained by counting the number of encodings that yield a particular delay value. Encodings under which the receivers are unable to decode all the packets are also taken into account. The probability that decoding will fail is given by

$$\mathcal{P}(\overline{D}) = \frac{1}{\mathcal{P}_o} \sum_{j=1}^{E} \mathcal{P}(\overline{D}|E_j) \cdot \mathcal{P}(E_j), \tag{2.23}$$

with $\mathcal{P}(E_j) = (1-e)^{N \cdot t_s - i} \cdot e^i$, $i \in 0 \ldots K$ and $\mathcal{P}(\overline{D}|E_j) = (1 - \frac{R_j}{R})$, where R_j is the number of encodings that guarantee successful decoding under the erasure pattern E_j and $R = \max_j R_j$. The corresponding values for the

cases of interest can be found in Table 2.1. As expected, the probability of unsuccessful decoding increases with the erasure probability.

2.4.2.4 Numerical Results

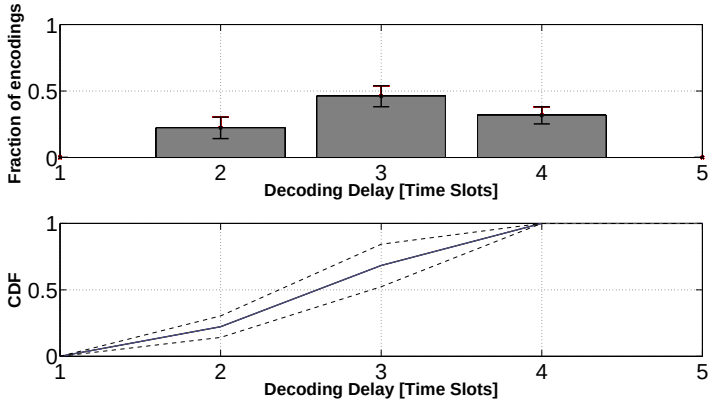
Delay Distribution

We now present histograms and cumulative functions for erasure patterns with 0.05, 0.1, and 0.2 erasure probability. The number of experiments used in the study are shown in Table 2.1. The histograms in Figures 2.9 and 2.11 correspond to the cases of $3N2M$ and $3N3M$, respectively. Since we are trying all possible encodings, we obtain a CDF for the delay. The dotted lines represent the confidence intervals that follow from the choice of erasure patterns, as described in the previous section. From Figure 2.9(a), for instance, we conclude that by randomly choosing an encoding we get a delay equal to 2 time slots with probability 0.22 ± 0.08 and equal to 3 time slots with probability 0.68. For the case of 3 packets and 0.05 erasure probability (Figure 2.11) we can observe that the probability of a decoding delay within 4 time slots is approximately 0.56.

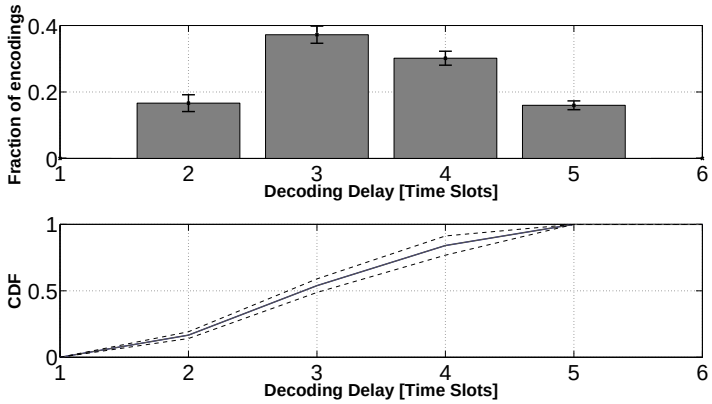
To illustrate the practical value of this brute-force analysis, we shall now consider a comparison between the distribution for encodings with only one uncoded packet with the distribution for completely uncoded transmission. Figure 2.12(a) and Figure 2.12(b) show the gap between the two cases. For instance, the case presented in Figure 2.12(a) can decode almost 0.55 of packets within 4 time-slots, whereas the instance depicted in Figure 2.12(b) can manage within the same number of time-slots almost 0.47 encodings. For the case of 2 packets and 0.2 erasure probability we can observe that sending only one uncoded packet within the encodings guarantees a delay within 4 time-slots with probability 0.60 (Figure 2.10(a)), whereas sending only uncoded packets has probability 0.50 of delay within 4 time slots (Figure 2.10(b)). Even if the number of packets is small, the analysis of network coding delay provides insight as to whether or not it is useful to code and what generation size one should use.

Distribution fitting

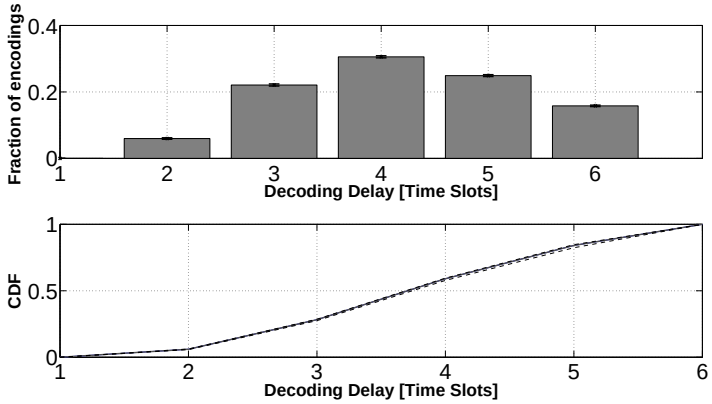
To find the best for the delay distribution among well studied and mathematically tractable probability distributions, we used standard curve fitting tools. It turns out that the delay distribution is generally well approximated by a normal distribution. The specific findings for the scenarios under consideration are given in Table 2.2. The cumulative distribution functions for $3N2M$ are given in Figure 2.13 for $e = 0.05$, $e = 0.1$, and $e = 0.2$ erasure



(a) $e = 0.05$

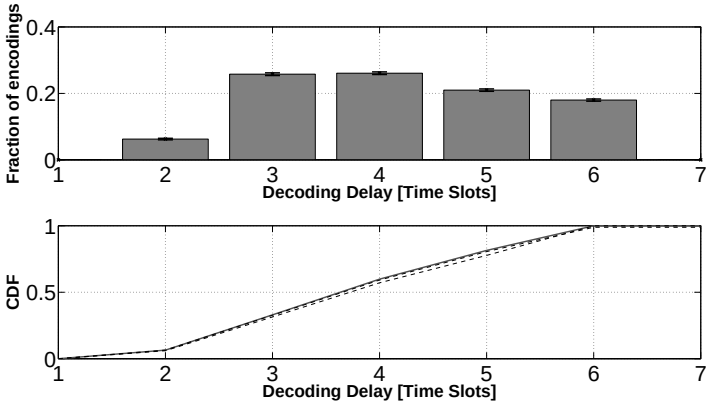


(b) $e = 0.1$

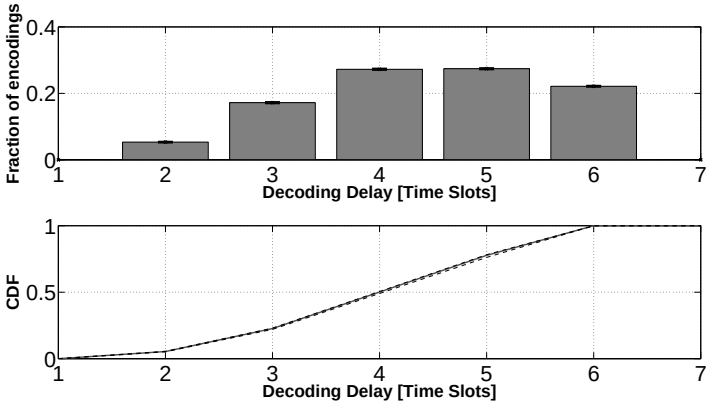


(c) $e = 0.2$

Figure 2.9: Histograms and cumulative functions for the case $3N2M$ and different erasure probabilities.



(a) $e = 0.2$ and one uncoded packet



(b) $e = 0.2$ and uncoded packets

Figure 2.10: Histograms and cumulative functions for the case $3N2M$, erasure probability $e = 0.2$ and encoding matrices with one uncoded packet within 6 slots and only uncoded packets within 6 slots.

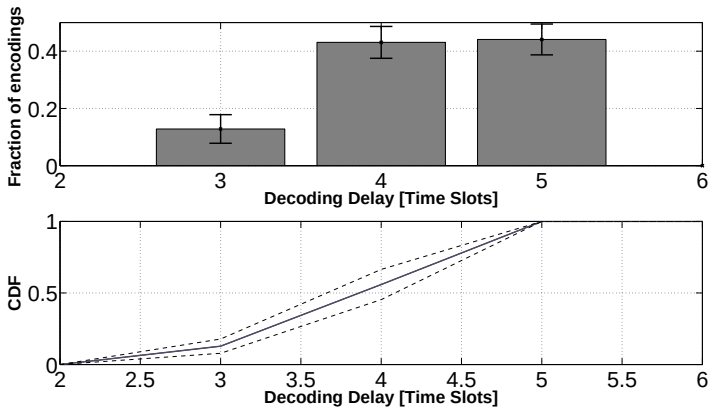


Figure 2.11: Histogram and cumulative function for the case $3N3M$ and erasure probability $e = 0.05$.

probability and for 95% confidence intervals. As we can see from the table and also from the figure, the best matching with the normal distribution is for the case of $e = 0.2$, where the number of erasure patterns is highest.

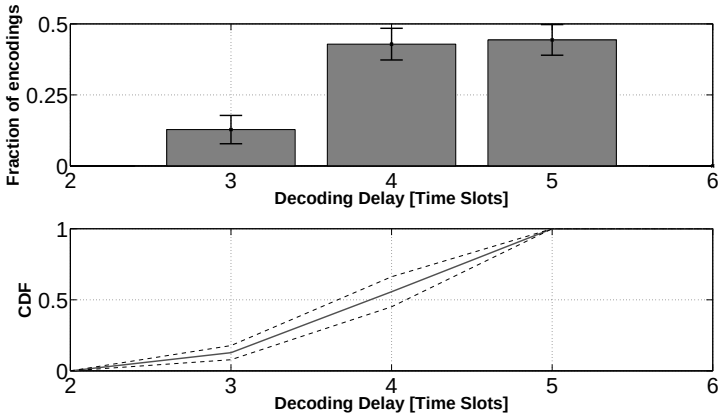
Table 2.2: Fitting the cases of interest with normal distribution

Experiment and e		Mean	Std. Err.[2]	Sigma	Std. Err.
3N2M	0.05	3.074	0.043	0.378	0.031
	0.1	3.447	0.031	0.471	0.022
	0.2	4.205	0.018	0.470	0.012
4N2M	0.05	3.077	0.043	0.382	0.031
	0.1	3.438	0.031	0.479	0.022
	0.2	4.003	0.025	0.385	0.022
2N3M	0.05	4.009	0.004	0.501	0.003
3N3M	0.05	3.458	0.008	0.950	0.005

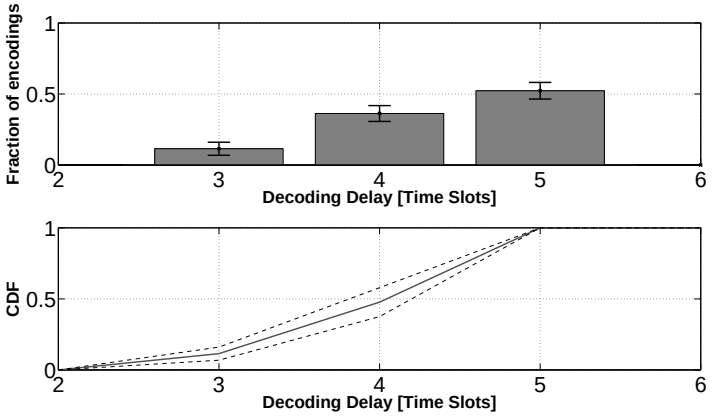
Ordered Delivery Delay

When transmitting linear combinations to multiple receivers, it is not always the case that the sent packets are decoded in the right order to be delivered to the application. It is thus reasonable to carry out a separate analysis to measure the ordered delivery delay illustrated in Table 2.3.

[2]Standard Error

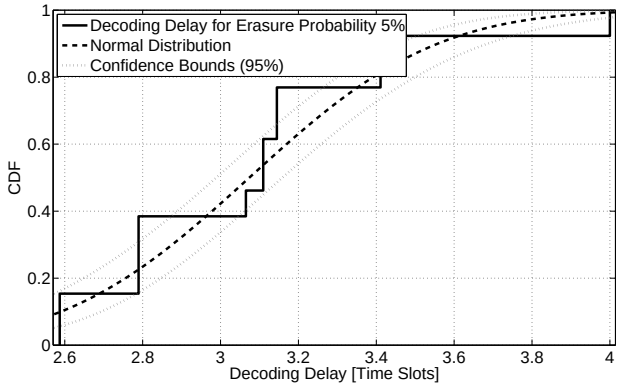


(a) $e = 0.05$ and one uncoded packet

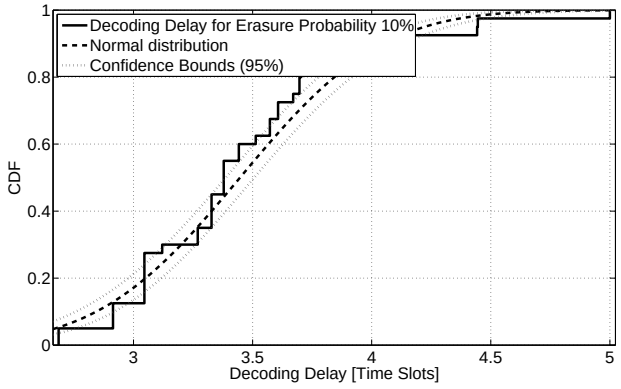


(b) $e = 0.05$ and uncoded packets

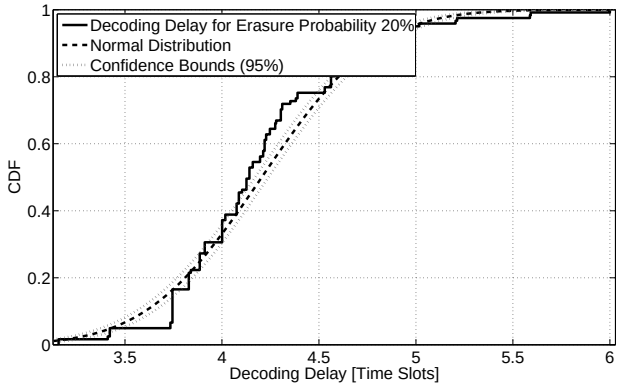
Figure 2.12: Histograms and cumulative functions for the case $3N3M$, erasure probability $e = 0.05$ and encoding matrices with one uncoded packet within 5 slots and only uncoded packets within 5 slots.



(a) $e = 0.05$



(b) $e = 0.1$



(c) $e = 0.2$

Figure 2.13: Fitting distribution for the case $3N2M$.

Table 2.3: Cases of interest for the ordered delivery delay

Experiment and e		Ordered delivery[%] min	max
3N2M	0.05	66.66	75.75
	0.1	66.66	75.23
	0.2	66.66	78.20
4N2M	0.05	66.66	75.75
	0.1	66.66	75.23
	0.2	66.66	79.16
2N3M	0.05	38.04	38.10
3N3M	0.05	38.01	47.03

In Table 2.3 we can see that the generation size has a significant impact on this performance metric. For example, for the cases $3N2M$ and $4N2M$ the percentage of packets that are obtained in-order varies between 66.66% and 79.16%, whereas for the cases $2N3M$ and $3N3M$ this percentage varies between 38.01% and 47.03%.

2.5 Concluding Remarks

Motivated by real-time applications with stringent deadlines, we considered the delay distribution of RLNC for multiple receivers. First, we derived the probability distribution for the delay incurred by RLNC over erasure channels for one receiver. Our results hold for small as well as large field sizes. The number of packets to encode can be arbitrary. Through a combinatorial approach, we were able to derive the probability distribution of the delay for RLNC, in the case of an erasure channel. We make use of this distribution to present some relevant behaviours of the delay exhibited by this coding technique. More precisely, we presented evidences that show that, even for a small finite field size, the probability of having M linearly independent combinations after M received packets is already close to 1. We have also showed that to obtain a similar performance to the standard ARQ scheme, one can use RLNC in a finite field of small size, without the need of a feedback channel. Second, by determining the delay distribution of RLNC for the case of two receivers, we were able to identify which parameter settings can meet specific worst-case guarantees. The benefits of RLNC in the broadcast scenario of interest were further highlighted by comparing its delay distribution

to that of three other transmission schemes. Perhaps the most important insights are that network coding over $\mathbf{GF}(2^4)$ for two receivers outperforms ARQ and that $\mathbf{GF}(2)$, although simple to implement, induces a heavy tail in the delay distribution. Third, given the mathematical difficulty in deriving an exact expression for $R > 1$, we resorted to brute-force analysis of all possible encodings on a statistically significant set of erasure patterns. Naturally, the maximum delay can be achieved be selecting a sufficiently large field size. However, this is not always an option. Specifically, using $\mathbf{GF}(2)$ has the advantage that no finite field multiplications are required, but only simple XOR operations. This reduces the decoding complexity as well as the overhead required for storing the coding coefficients and decoding matrices. A prime example application where this is important is P2P file distribution, where the sheer size of data requires efficient coding and decoding algorithms. Another example are sensor network applications where overhead is critical. Any increase in packet header size significantly reduces the portion of the packet available for data, due to the very small packet sizes, and any substantial increase in memory footprint runs the risk of having to use very energy consuming flash memory, rather than efficient RAM for the decoding matrices.

Our analysis was given for the case of one-hop communication, but the insights we obtained can be translated to more complex networks. In fact, the sender can be either the original source at the edge of the network or an intermediate node somewhere in the core. Likewise, the receivers can be either intermediate nodes in the network or the final destination at the edge of the network. On the other hand, the packets as defined in our problem statement could in fact be coded packets if the sender acts as an intermediate node.

It is important to note that the brute-force methodology to compute the delay distribution can be used not only for RLNC but also for other coding schemes. One example thereof is uncoded packets, whose performance we compared against the case with coding. It would be interesting to understand if the delay for random network coding with some constraints (for example, in the choice of coefficients) is still well approximated by a normal distribution. Also worth noting is the different results obtained for delay and ordered delivery delay. We believe both metrics are important, depending on the particular application.

What does not change is the fact that computing the delay distribution of network coding is a high-dimensional and computationally demanding problem. Since we are interested in guidelines for system design, the proposed methods based on a Markov chain model and a brute-force approach could be used in combination to seek meaningful heuristics and bounds for the de-

sign. In particular, a hybrid search would be able to select the most relevant erasure patterns while capitalizing on the Markov chain to take the impact of the field size into account.

Chapter 3

Data Collection in Large Scale Networks

3.1 Motivation

Although the previous chapter focused on the delay distribution of RLNC using intra-flow coding, characterizing the transmission time in the presence of multiple data flows, i.e., inter-flow network coding, is also critical. This chapter addresses the protocol design for the data collection applications in multi-hop networks, where the performance metric is the minimization of the time required to gather all data packets from the network.

Efficient data gathering is a key to enabling a variety of applications, e.g., mobile networks, monitoring in smart grids, collection of environmental data, sensing in smart cities. Usually, data gathering is solved by using unicast transmission protocols based on the direct diffusion method [64] or expected number of transmissions [65] and multicast transmission using Luby Transform codes [66], but the variety of the applications and the network type can change the performance of the protocol. As a particular example, smart grid applications need to collect the data at various levels, from the lowest one in the low-voltage medium, e.g., the houses (or smart meters), to the higher level in the medium-voltage medium, e.g., the central unit data collection. The density of the nodes also increases significantly from the top to the base. The data size can be small or large, less critical (e.g., common reading of the smart meters) or critical (e.g., more demanding informations about the state of the network, i.e., the current/voltage level, phases). Thus, in general, the data collection protocol needs to consider various challenges imposed by the network topology and size, the length of the data and, in some cases, the deadline type.

The potential benefits of network coding have been introduced for data gathering protocols. Hence, [35] proposes an algorithm for the wireless sensor networks based on two schemes, i.e., RLNC and a systematic approach. The provided results demonstrate the reliability of network coding for data collection protocols for a 6 x 6 nodes in a grid network using a field size $\mathbf{GF}(2^4)$. More recently, [36] shows that significant improvements can be achieved for data gathering applications in smart grids when comparing to a well-known master-slave protocol by using a simple, but efficient protocol based on network coding approach. This protocol is based on TSNC [37], meaning sending sparse codes at the beginning of the transmission process and more dense towards the end. The sparsity is given by using a systematic transmission approach and feedback mechanisms. But, the performance evaluation is provided for some particular constraints of the protocol, e.g., the sparsity of the coded packet and the feedback type are fixed, the number of packets sent by each node is limited to one, the field size for performing network coding operations is two.

Hence, on the one side, the data gathering protocols need to embrace several challenges imposed by the network and the application. On the other side, the previous results for data collection protocols using network coding are restricted in the sense that they use a specific topology or particular parameters, e.g., for the sparsity level, the feedback type, the field size, the number of data packets. Thus, we set out to understand the feasibility of RLNC and TSNC for data gathering protocols, considering challenged networks and in the presence of diverse feedback techniques. Inspired in part by the idea introduced in [36], we make the following contributions:

- *Mathematical analysis*: We propose a model that describes the packet flows between nodes using virtual queues. Since the results in [36] are valid only for $\mathbf{GF}(2)$, this model applies for any field size, number of nodes, and number of data packets, and a line network. The findings are valid for dense codes, i.e., common RLNC, simple sparse codes and sparse codes including feedback techniques, i.e., tuned versions of RLNC.

- *Design optimization*: We enhance the performance of the TSNC protocol, by relaxing the conditions from [36]. The gains for TSNC under various sparsity levels, the implicit feedback imposed by overhearing the transmissions between the neighbour nodes, and the effect of the explicit feedback sent deliberately by a node, are some of the purposes to optimize the protocol design. This part is valid for an arbitrary network topology, any number of nodes and number of data packets, but only for Galois field two, i.e., $\mathbf{GF}(2)$.

- *Communication strategies*: We compare the gains of the TSNC protocol with some existing protocols, e.g., master slave, conventional RLNC, and with some tuned versions of RLNC. We aim to understand the gain differences and under which conditions one protocol outperforms the other.

The remainder of the chapter is organized as follows. Section 3.2 offers the related work. The queueing model for the line network is given in Section 3.3. The numerical model for the grid network is provided in Section 3.4. The numerical results are given in Section 3.5 and the chapter concludes with Section 3.6.

3.2 Related Work

Data collection protocols have been deployed in [33, 34]. Ref. [33] uses a network coding protocol based on retransmissions, while [34] provides an approach using network coding and duty-cycling in sensor networks. But, network coding protocols are suitable for broadcast applications and, thus, in multi-hop networks overhearing the transmissions by the neighbour nodes creates redundant packets. This increases the availability of the data packets. In this sense, [67] shows the trade-off between the transmission of redundant packets and the amount of overhearing in wireless sensor networks. But, the overhearing is limited to only two neighbour nodes. Moreover, [35] provides an algorithm based on overhearing and uses either a common RLNC or a systematic approach, over the field size $\mathbf{GF}(2^4)$. The algorithm imposes for each node to store only 10% of the packets it overhears. The results are implemented in wireless sensors using a grid network with 36 nodes and show that the end-to-end packet error rate can be reduced for highly dynamic environments. Furthermore, the authors in [36] show the benefits of using network coding in terms of completion time in a large scale smart grid scenario, where the protocol is based on TSNC. The level of sparsity is also exploited by using different feedback techniques. But, the number of packets used in combinations is limited, the feedback frequency is fixed, the field size is two and just one data packet is initially available for transmission at each node.

For the queueing analysis, some previous works have been focused on routing and scheduling in multi-hop networks. The well-known work by Tassiulas *et al.* in [68] introduces the back-pressure approach, where the downstream nodes push back and slow down the flow coming for the upstream nodes by maximizing the queue-weighted sum of the rates. This approach is throughput optimal. Based on this scheme, [69] proposes the back-pressure

method using network coding in the context of multicast transmissions. This work uses intra-flow network coding, while our work addresses the problem of inter-flow network coding.

3.3 Analysis for the Line Network

We first provide a mathematical model for a line network with N nodes, each one wants to send M data packets to a base station (BS). The channels are described by the erasure probability e_i, $\forall i = \{1, 2, ..., N\}$, where the nodes are numbered from the one farthest away from the BS to the one closest to it. The transmission probability for each node i is p_i, $\forall i = \{1, 2, ..., N\}$, where a node i accesses at random the channel for transmission. A transmission is considered without collision if node i transmits and node $i + 1$ and $i + 2$ do not transmit. Time is slotted, one packet is transmitted in a time slot, and nodes have a half-duplex constraint. We assume that RLNC operations are performed, i.e., a node mixes the packets it has in its queue (original source packets and coded packets) with coefficients selected uniformly at random from the Galois field $\mathbf{GF}(q)$. The density of the coded packets can be controlled by performing combinations using a limited number of packets (C) selected uniformly at random from the available ones. Thus, a coded packet is C-sparse if it contains C non-zero coefficients from $\mathbf{GF}(q)$. Moreover, a coded packet is innovative if it is linearly independent from the other packets in the queue. The process stops when the BS collects all the packets from the nodes in the network. Thus, the completion time is defined by the total time required to gather all the data packets from the network at the BS.

The following definitions are useful for the rest of the chapter.

Definition 8. *The queue of innovative packets for a node i at a given time t is represented by $Q_i(t)$.*

Definition 9. *The queue of non-innovative packets for a node i at a given time t is defined by $\overline{Q}_i(t)$.*

Definition 10. *The total queue including the innovative packets and the non-innovative packets for a node i at a given time t is defined by $Q_i^o(t) = Q_i(t) + \overline{Q}_i(t)$.*

Definition 11. *The density of innovative packets for a node i at a given time t is characterized by $\frac{Q_i(t)}{Q_i^o(t)}$.*

Definition 12. *The probability of transmitting with success a packet by a node i is obtained from the joint of the conditions: (1) the channel does not*

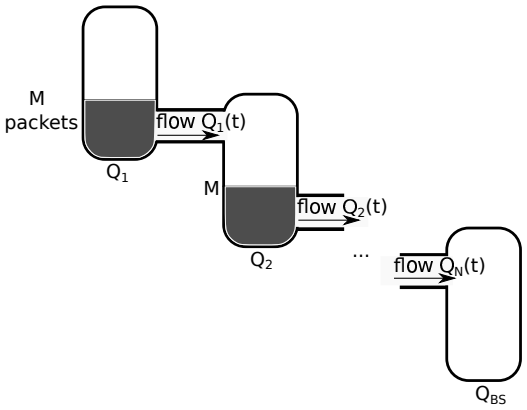


Figure 3.1: Coupled queues of innovative packets, initial queue size at each node is M packets and at BS is 0 packets.

erase the packet $(1 - e_i)$ *and (2) no collision occurs* $p_i \cdot (1 - p_{i+1}) \cdot (1 - p_{i+2})$. *Thus, we define the parameter* $K_i = (1 - e_i) \cdot p_i \cdot (1 - p_{i+1}) \cdot (1 - p_{i+2})$.

3.3.1 General Case

We model the problem by considering coupled queues of innovative packets, one queue for each node. The meaning is similar as in the case of coupled tank system, where the idea is to control the liquid level in tanks and the flow between tanks. Here, the liquid level is represented by the number of innovative packets and the flow of liquid between tanks is equivalent to the flow of innovative packets. The main concept is based on the changes in time with the innovative packets for each queue. We assume that the initial size of each node's queue with innovative packets is M, $Q_i(0) = M$, and the BS's queue is 0, $Q_{BS}(0) = 0$. These are virtual queues. The changes in the queues are represented by the flows of innovative packets between two nodes, as Figure 5.1(a) shows. The flows can be different for each node depending on the characteristics of the channel (e.g., erasure probability), the application feature (e.g., transmission probability), and the protocol requirements (e.g., coding method, feedback). The meaning of the flow of innovative packets is provided by the following definition.

Definition 13. *The flow of innovative packets for a node* i *at a given time* t *is represented by the probability of transmitting with success an innovative packet* $\mathcal{P}_i(t)$.

Proposition 1. *In general, for each node* i, *the description of the innovative packets queue variation in time is provided by the input flow of innovative*

packets that comes from the previous node, $\mathcal{P}_{i-1}(t)$, and the flow of innovative packets that leaves node i at time t, $\mathcal{P}_i(t)$, while for the BS, the flow is given by the last node, N,

$$\begin{aligned} \frac{dQ_i(t)}{dt} &= \mathcal{P}_{i-1}(t) - \mathcal{P}_i(t), \\ \frac{dQ_{BS}(t)}{dt} &= \mathcal{P}_N(t), \end{aligned} \tag{3.1}$$

where $\mathcal{P}_0(t) = 0$, $\forall i = 1, 2, ..., N$.

The completion time T is given when $N \cdot M$ packets are received at the BS, $Q_{BS} = M \cdot N$, meaning that the queues of innovative packets at each node become zero. We provide the completion time analysis for the following cases. First, a node performs RLNC operations using all the packets in its queue. Here, the model is based only on the evolution of the innovative packets at each node. Second, the number of packets used to perform RLNC operations are limited using the C parameter and this corresponds to the sparse network coding case or sparsity case. In this case, the model focuses on both the queue of innovative and non-innovative packets. Then, we provide insights when feedback techniques are applied in the network. The method uses again the queue of innovative and non-innovative packets. Thus, we start with common RLNC operations, but then, when using different sparsity levels and sparsity with diverse feedback techniques, we deal with tuned versions of RLNC. The difference between the pure RLNC, the sparsity and the feedback cases is given by the flow of innovative packets between the queues, $\mathcal{P}_i(t)$, which is described by the following lemmas.

3.3.2 Analysis for the Pure RLNC Case

The probability of transmitting with success an innovative packet by a node i at time t is obtained from the joint of the conditions of transmitting with success a packet and that packet being innovative $(1 - q^{-Q_i(t)})$ [57].

Lemma 6. *The probability of transmitting with success an innovative packet from node i using field size $\mathbf{GF}(q)$ is*

$$\mathcal{P}_i(t) = K_i \cdot (1 - q^{-Q_i(t)}). \tag{3.2}$$

Corollary 1. *When the field size is high enough such that the obtained network coding combinations are linearly independent with high probability, thus,*

the completion time is

$$T = \max\left(\frac{i \cdot M}{K_i}\right), \forall i = \{1, 2, .., N\}. \tag{3.3}$$

Proof. We define τ_i such that the completion time for each node is given when $Q(\tau_i) = 0$, $\forall i = \{1, 2, .., N\}$. We start with the general description of the innovative packets flow variation given by (3.1). For the first node, we have $\frac{dQ_1(t)}{dt} = 0 - \mathcal{P}_1(t)$, where $\mathcal{P}_1(t) = K_1, \forall t \in [0, ..., \tau_1]$ and τ_1 is given for $Q_1(\tau_1) = 0$. We obtain $Q_1(t) = C_1 - K_1 t \mid_0^{\tau_1}$. We know that $Q_1(0) = M$ and get $C_1 = M$, hence, $\tau_1 = \frac{M}{K_1}$. Thus, $Q_1(t) = M - \begin{cases} K_1 t, & 0 \leq t \leq \frac{M}{K_1} \\ 0, & \text{others.} \end{cases}$ Then, the completion time for the first node is given by $T_1 = \frac{M}{K_1}$. For the second node we proceed similarly as for the first node. Here, we distinguish three cases:

$$\frac{dQ_2(t)}{dt} = \begin{cases} K_1 - K_2, & t \leq \tau_1 \text{ and } t < \tau_2 \\ K_1, & t \leq \tau_1 \text{ and } t > \tau_2 \\ K_2, & t > \tau_1 \text{ and } t \leq \tau_2 \\ 0, & \text{others.} \end{cases} \tag{3.4}$$

The first two cases correspond to the situation when $t \leq \tau_1$, then $\tau_1 > \tau_2$, which means that the completion time is given by τ_1, previously defined. For the third case, we get

$$Q_2(t) = M - K_2 t + K_1 \tau_1, \quad t > \tau_1 \text{ and } t \leq \tau_2. \tag{3.5}$$

Thus, for $Q_2(\tau_2) = 0$, we get $\tau_2 = \frac{2M}{K_2}$. The completion time for the second node is given by $T_2 = \frac{2M}{K_2}$.

For the other nodes, the same reasoning is applied and for the general case, the completion time for node i is $T_i = \frac{i \cdot M}{K_i}$. Then, the completion time of the system is provided by the node that takes longer the transmission of the packets, which concludes with (3.3). □

Remark 1. *Regarding the transmission probabilities, if we equalize the terms from the expression (3.3), i.e.,* $\frac{M}{K_1} = \frac{2 \cdot M}{K_2} = ... = \frac{N \cdot M}{K_N}$*, then* $\frac{K_i}{K_{i+1}} = \frac{i}{i+1}$*, the optimum transmission probabilities to minimize the completion time are*

$$p_{N-j} = \frac{A \cdot p_{N-(j+1)}}{1 + A \cdot p_{N-(j+1)} - p_{N-(k-2)}}, \tag{3.6}$$

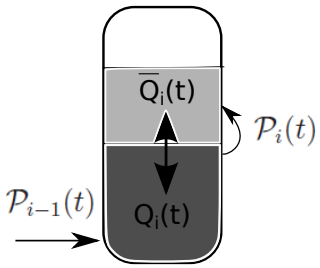


Figure 3.2: Sparsity description for node i.

where

$$A = \frac{N-j}{N-(j+1)} \cdot \frac{1-e_{N-(j+1)}}{1-e_{N-j}}, \textit{ with } j = 0, 1, ..., N-1$$
$$\textit{and } k = \begin{cases} j, & \textit{if } j \geq 2. \\ \nexists, & \textit{otherwise} \end{cases} \tag{3.7}$$

3.3.3 Analysis for the Sparse Network Coding Case

We address the sparsity case including the flow of innovative and non-innovative packets. Two approaches are considered. The first considers the sparsity of a coded packet, called C-sparse packet, and the second is based on the density of a sparse packet.

C-Sparse Packet

Considering that the queue of a node can contain innovative as well as non-innovative packets, we introduce the influence of using sparse coded packets. The representation is given in Figure 3.2. The idea is to model the evolution of the density of innovative packets. The speed of flow $\mathcal{P}_i(t)$ depends on the density of the innovative packets, meaning that the higher is the density, the faster is the flow.

Lemma 7. *The probability of transmitting with success at least one innovative packet from node i using C-sparse coded packets and field size* $\mathbf{GF}(q)$ *is*

$$\mathcal{P}_i(t) = K_i \cdot \sum_{j=1}^{\min(C, Q_i(t))} (1 - \frac{1}{q^j}) \cdot \frac{\binom{Q_i(t)}{j} \cdot \binom{Q_i^o(t) - Q_i(t)}{C-j}}{\binom{Q_i^o(t)}{C}}, \tag{3.8}$$

where $C \leq Q_i^o(t)$.

Proof. The idea comes from the analogy to calculate the probability of drawing balls from an urn without replacement. An urn has a white balls and b black balls, and we extract $n \leq a+b$ balls without replacement, then, the probability to have c white balls, for $c \leq a$, is

$$\mathcal{P}_{a,b}^{c,n-c} = \frac{\binom{a}{c} \cdot \binom{b}{n-c}}{\binom{a+b}{n}}. \tag{3.9}$$

In our case, the white balls are the innovative packets and the black balls are the non-innovative packets. We count for the total possible cases of having innovative packets, meaning the summation $\sum_{j=1}^{\min(C,Q_i(t))} \frac{\binom{Q_i(t)}{j} \cdot \binom{Q_i^o(t)-Q_i(t)}{C-j}}{\binom{Q_i^o(t)}{C}}$. This expression means that exactly j packets are innovative out of C, given that $j = 1, 2, ..., \min(C, Q_i(t))$. Then, the probability of transmitting with success at least one innovative packet is given by (3.8). □

We approximate the expression (3.8) using the following lemma.

Lemma 8. *The probability of transmitting with success at least one innovative packet from node i using C-sparse coded packets and field size* $\mathbf{GF}(q)$ *under the assumptions of Lemma 7 is*

$$\mathcal{P}_i(t) \geq K_i \cdot (1 - \frac{1}{q}) \cdot \left(1 - \left(1 - \frac{Q_i(t)}{Q_i^o(t)}\right)^C\right). \tag{3.10}$$

Proof. We use the total queue $Q_i^o(t)$ and the non-innovative queue $\overline{Q}_i(t)$. The probability of having all C packets non-innovative is provided by the total number of possible combinations when C packets are selected, given that the queue has $\overline{Q}_i(t)$ non-innovative packets and a total size of $Q_i^o(t)$, i.e., $\frac{\binom{\overline{Q}_i(t)}{C}}{\binom{Q_i^o(t)}{C}}$, where $C \leq \overline{Q}_i(t)$. Using Pascal's triangle, we know that $\binom{a}{b} = \prod_{j=0}^{b} \frac{a+j}{j}$, then, we get $\frac{\binom{\overline{Q}_i(t)}{C}}{\binom{Q_i^o(t)}{C}} = \prod_{j=1}^{C} \frac{\overline{Q}_i(t)+j}{Q_i^o(t)+j}$. Assuming that $Q_i^o(t) \geq \overline{Q}_i(t)$, we approximate $\frac{\binom{\overline{Q}_i(t)}{C}}{\binom{Q_i^o(t)}{C}} = \left(\frac{\overline{Q}_i(t)}{Q_i^o(t)}\right)^C$. Hence, $\left(\frac{\overline{Q}_i(t)}{Q_i^o(t)}\right)^C = \left(\frac{Q_i^o(t)-Q_i(t)}{Q_i^o(t)}\right)^C = \left(1 - \frac{Q_i}{Q_i^o(t)}\right)^C$. Then, the probability of transmitting with success at least

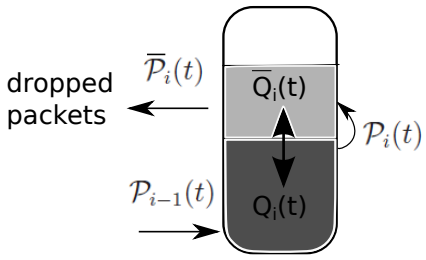


Figure 3.3: Sparsity description including feedback for node i.

one innovative packet using a field size $\mathbf{GF}(q)$ is $\mathcal{P}_i(t) \geq K_i \cdot (1 - \frac{1}{q}) \cdot \left(1 - \left(1 - \frac{Q_i(t)}{Q_i^o(t)}\right)^C\right)$. □

Density of a C-Sparse Packet

As an alternative method, we consider the previous results from [70], where the density ρ of a C-sparse packet is given by

$$\rho = \frac{C}{Q_i^o(t)} \leq 1 - \frac{1}{q}. \tag{3.11}$$

Maintaining ρ constant means increasing C as the size of the node's queue grows, while maintaining the sparsity C constant means changing the density in time.

Lemma 9. *According to [70] (Theorem 1), the probability of transmitting with success an innovative coded packet, which has density ρ, from node i using field size* $\mathbf{GF}(q)$ *is*

$$\mathcal{P}_i(t) \geq K_i \cdot (1 - (1 - \rho)^{Q_i(t)}). \tag{3.12}$$

3.3.4 Feedback Implications

The feedback provides information about the received packets, thus, sending any type of feedback requires updating the non-innovative packets in the queue of the node that transmitted the packets. We distinguish the following situations.

3.3.4.1 Pure RLNC Case

According to equation (3.2), there is no dependency on the non-innovative packets, then, the feedback brings no benefit in terms of the completion time. The only case when the feedback can be useful here is if the transmission probabilities need to be updated during the transmission process. For example, let us assume we have a line network with $N = 2$ and node 2 finishes faster the transmission of its data packets. Then, the feedback sent by the BS can help to minimize the transmission probability of node 2, such that node 1 can also finish faster its transmissions.

3.3.4.2 Sparse Network Coding Case

According to equation (3.10) and (3.12), the expressions depend on the total size of the node's queue, including the innovative and non-innovative packets. The sparsity without any feedback represents a particular case, because the flow of non-innovative packets is considered 0, as Figure 3.2 shows comparing with Figure 3.3. The level of sparsity is influenced by these two types of feedback, i.e., by reducing the non-innovative packets from the nodes's queue, the coded packets become denser. Thus, controlling the level of sparsity by using a feedback might be critical, because denser coded packets can have an impact on the encoding complexity and the efficiency of the decoders. The description of the non-innovative packets flow is provided by the following definition.

Definition 14. *The flow of non-innovative packets for a node i at a given time t is represented by the probability of not transmitting with success an innovative packet $\overline{\mathcal{P}}_i(t)$.*

Proposition 2. *In general, for each node i, the description of the non-innovative queue variation in time is provided by the input flow of innovative packets of node i, $\mathcal{P}_i(t)$, and the flow of non-innovative packets that leaves node i at time t, $\overline{\mathcal{P}}_i(t)$, while for the BS, there is no flow for the non-innovative packets,*

$$\frac{d\overline{Q}_i(t)}{dt} = \mathcal{P}_i(t) - \overline{\mathcal{P}}_i(t), \tag{3.13}$$

$\forall i = 1, 2, ..., N$.

We explain separately the two types of feedback, e.g., the implicit and the explicit feedback. Both of them are described from the side of the node that receives the feedback. In general, the non-innovative packets of node i are

already in the queue of node $i+1$ and thus, for both feedback schemes, we focus on modelling the changes that appear for the non-innovative packets at node i. We keep the same idea as for the packet transmission, hence, the feedback transmitted by node $i+1$ is heard by node i if node i and node $i-1$ do not transmit. The difference between the implicit and explicit feedback is given by the description of the non-innovative packet flow.

Explicit Feedback

In this case, BS spreads to the nodes the information about all received packets. The feedback is flooded to all the nodes in the network, which means we have an inverse flow of transmission from the BS to node 1. A node $i+1$ sends the explicit feedback to a node i, including also the information about all the non-innovative packets of node i. The feedback frequency can vary and the time spent to send the explicit feedback is also counted to the total completion time.

We define the parameter for the innovative packet flow $K_i^{ef} = K_i \cdot (1-p_f)$ and the non-innovative packet flow $\overline{K}_i^{ef} = (1-e_i) \cdot p_{i+1} \cdot p_f \cdot (1-p_i) \cdot (1-p_{i-1})$, for node i. The probability of transmitting an explicit feedback is defined by p_f and represents the percentage of time allocated for the feedback from the transmission probability p_i for node i. The higher is p_f, the more often the feedback is sent.

Lemma 10. *The description for a node i receiving an explicit feedback from a downstream node $i+1$ includes both the flow of innovative packets and the one with non-innovative packets, where $\mathcal{P}_i(t)$ and $\overline{\mathcal{P}}_i(t)$ are defined by*

$$\begin{aligned} \mathcal{P}_i(t) &= K_i^{ef} \cdot (1-\frac{1}{q}) \cdot \left(1-\left(\frac{\overline{Q}_i(t)}{\overline{Q}_i(t)+Q_i(t)}\right)^C\right), \\ \overline{\mathcal{P}}_i(t) &= \overline{K}_i^{ef} \cdot \overline{Q}_i(t). \end{aligned} \tag{3.14}$$

Proof. The expression for the innovative packet flow is similar with the one in (3.10), but here it is written in terms of $\overline{Q}_i(t)$. The proof for the non-innovative packet flow is given by simply following Figure 3.3. Since the explicit feedback provides information about all the non-innovative packets, the non-innovative packets dropped are clearly described by the flow $\overline{K}_i^{ef} \cdot \overline{Q}_i(t)$. The flow of non-innovative packets $\overline{\mathcal{P}}_i(t)$ is not a probability in this case, but still contains value between 0 and 1. This happens because the variation of the non-innovative packets has the meaning of a low pass filter, $\mathcal{P}_i(t) - \overline{\mathcal{P}}_i(t) = \frac{d\overline{Q}_i(t)}{dt}$, which means $\frac{1}{K_i^{ef}}\overline{\mathcal{P}}_i(t) - \overline{Q}_i(t) = \frac{d\overline{Q}_i(t)}{dt}$. The input

is $\frac{1}{\overline{K}_i^{ef}}\mathcal{P}_i(t)$ and the output is $\overline{Q}_i(t)$. Since $\frac{1}{\overline{K}_i^{ef}}\mathcal{P}_i(t) \in \{0, ..., \frac{1}{\overline{K}_i^{ef}}\}$, then, $\overline{\mathcal{P}}_i(t) = \overline{K}_i^{ef} \cdot \overline{Q}_i(t) \in \{0, ..., 1\}$. □

Implicit Feedback

This feedback appears if a node i updates its buffer status when hearing the transmission from node $i+1$. Thus, if packets of node i are already in the buffer of node $i+1$, then, node i removes those packets in the future combinations. We use the same parameter for the innovative packet flow as for the general case, K_i, and for the non-innovative packet flow we define $\overline{K}_i^{if} = (1-e_i) \cdot p_{i+1} \cdot (1-p_i) \cdot (1-p_{i-1})$.

Lemma 11. *The description for a node i hearing the transmission from node $i+1$ includes both the flow of innovative packets and the one with non-innovative packets, where $\mathcal{P}_i(t)$ and $\overline{\mathcal{P}}_i(t)$ are defined by*

$$\mathcal{P}_i(t) = K_i \cdot (1 - \frac{1}{q}) \cdot \left(1 - \left(\frac{\overline{Q}_i(t)}{\overline{Q}_i(t) + Q_i(t)}\right)^C\right) \tag{3.15}$$

$$\overline{\mathcal{P}}_i(t) = \overline{K}_i^{if} \cdot \left(1 - \left(\frac{M}{M + \overline{Q}_i(t)}\right)^C\right).$$

Proof. The expression for the innovative packet flow is similar with the one in (3.10), but written in terms of $\overline{Q}_i(t)$. For the non-innovative packet flow, we based on the fact that the non-innovative packets from node i are already at node $i+1$. Thus, any transmission by node $i+1$ heard at node i is seen as a decreasing of the non-innovative queue size by $\overline{Q_i}(t)$ at node i. At node $i+1$, the probability of using C-sparse packets out of M packets is written as $\left(\frac{\binom{M}{C}}{\binom{M+\overline{Q}_i(t)}{C}}\right)^C = \left(\frac{M}{M+\overline{Q}_i(t)}\right)^C$, given that $\overline{Q}_i(t)$ packets are received from $Q_i(t)$. Hence, at node i, the probability that exists at least one non-innovative packet from node $i+1$ using C-sparse packets is $1 - \left(\frac{M}{M+\overline{Q}_i(t)}\right)^C$. □

3.3.5 Completion Time Performance

Our performance metric, the mean completion time, is computed from the variation of the number of innovative packets at BS. Thus, given $Q_{BS}(t)$,

we divide it by the total number of packets in the network and obtain the cumulative distribution function (CDF), $\frac{Q_{BS}(t)}{N \cdot M} = F_{BS}(t)$. Then, by deriving $F_{BS}(t)$, we get the probability density function (PDF), denoted by $f_{BS}(t)$. Using $f_{BS}(t)$, the expected value is $E = \int_1^{t_{BS}} t \cdot f_{BS}(t)dt$, where t_{BS} is the time limit over which are performed the results. The mean completion time is obtained from $T = \frac{N \cdot M}{f_{BS}(E)}$.

3.3.6 Solution for Two Nodes

Pure RLNC Case

For two nodes that perform common RLNC operations, the solution is derived from expression (3.2) and is

$$\frac{dQ_1(t)}{dt} = -K_1(1 - q^{-Q_1(t)}) \tag{3.16}$$

The result for $Q_1(t)$ is given by

$$Q_1(t) = \frac{\ln(1 - e^{C_1 \cdot \ln q - K_1 t \cdot \ln q})}{\ln q}, \tag{3.17}$$

where C_1 can be easily found from the condition $Q_1(0) = M$, which gives $C_1 = \log_q(1 - q^M)$. Substituting C_1 in the $Q_1(t)$, we obtain the following expression:

$$\begin{aligned} Q_1(t) &= \frac{\ln(1 - e^{\ln(1-q^M) - K_1 t \cdot \ln q})}{\ln q} \\ &= \frac{\ln(q^{-K_1 t} \cdot (q^{K_1 t} + q^M - 1))}{\ln q}. \end{aligned} \tag{3.18}$$

The derivative of $Q_1(t)$ is

$$\frac{dQ_1(t)}{dt} = -\frac{K_1 \cdot q^M}{q^{K_1 t} + q^M - 1}. \tag{3.19}$$

For the second node, we proceed similarly, thus,

$$\frac{dQ_2(t)}{dt} = \frac{K_1 \cdot q^M}{q^{K_1 t} + q^M - 1} - K_2(1 - q^{-Q_2(t)}). \tag{3.20}$$

$$Q_2(t) = \log_q \frac{C_2 \cdot q^{t(K_1-K_2)} - \frac{K_2\cdot(q^M-1)}{K_1-K2} + q^{K_1 t}}{q^M - 1 + q^{K_1 t}} \tag{3.22}$$
$$= \log_q \frac{(q^{2M}-1)\cdot q^{t(K_1-K_2)} + \frac{K_2\cdot(q^M-1)}{K_1-K_2}\cdot(q^{t(K_1-K_2)}-1) + q^{K_1 t}}{q^M - 1 + q^{K_1 t}}$$

The solution for $Q_2(t)$ is given in (3.22). Using the same procedure as for Q_1, we find the constant C_2 as being

$$C_2 = \frac{(q^{2M}-1)}{\ln q} + \frac{K_2\cdot(q^M-1)}{(K_1-K_2)\cdot \ln q}. \tag{3.21}$$

Remark 2. *For the particular case when $K_1 = K_2$, we have the following result:*

$$Q_2(t) = \log_q \frac{q^{2M} - 1 + K_2\cdot(q^M-1)\cdot \ln q + q^{K_1 t}}{q^M - 1 + q^{K_2 t}}, \tag{3.23}$$

since $\lim_{K_1-K_2\to 0^+} \frac{q^{t(K_1-K_2)}-1}{K_1-K_2} = \ln q$, *with* $t > 0$.

Sparse Network Coding Case

For the sparsity case, we provide the solution for two nodes using expression (3.12),

$$\frac{dQ_1(t)}{dt} = -K_1\cdot(1-(1-\rho)^{Q_1(t)}) \tag{3.24}$$

$$\frac{dQ_2(t)}{dt} = \frac{K_1\cdot(1-\rho)^{-M}-1)\cdot(1-\rho)^{K_1 t}}{((1-\rho)^{-M}-1)\cdot(1-\rho)^{K_1 t}+1}$$
$$-K_2(1-(1-\rho)^{Q_2(t)}). \tag{3.25}$$

Example 1. *When using expression (3.12), for specific values of the parameters, e.g., $K_1 = K_2 = \frac{1}{2}$ (might be $e_1 = e_2$ and $p_1 = 1$ and $p_2 = 0.5$) and $q = 2$, we have the results for $Q_1(t)$ in (3.26) and for $Q_2(t)$ in (3.27),*

$$Q_1(t) = -\frac{\ln(((1-\rho)^{-M}-1)\cdot(1-\rho)^{\frac{t}{2}}+1)}{\ln(1-\rho)} \tag{3.26}$$

$$Q_2(t) = \frac{1}{2}\cdot \tag{3.27}$$
$$\cdot\frac{-t\ln(1-\rho)+2\ln\frac{2(1+\frac{(1-\rho)^{\frac{t}{2}}}{(1-\rho)^M}-(1-\rho)^{\frac{t}{2}})}{((1-\rho)^M-1)\cdot t\ln(1-\rho)+2(1-\rho)^{(M-\frac{t}{2})}-\frac{2(-1+((1-\rho)^M)^2)}{(1-\rho)^M}}+2M\ln(1-\rho)}{\ln(1-\rho)}.$$

3.4 Analysis for the Grid Network

Given that the proposed model in the previous section is valid only for a line network, we extend the work to a grid network using numerical analysis and field size $\mathbf{GF}(2)$. Here, we consider a broadcast wireless network with N nodes, where each node wants to transmit M packets to a BS. The position of the nodes is randomly generated within a predefined square area. The information data can be lost with erasure probability e. The transmission range is defined by R. Thus, two nodes can transmit simultaneously only if the transmission areas defined by R do not intersect. Moreover, the nodes closer to the BS are called downstream nodes, and the one longer are the upstream nodes. Nodes have information about the downstream nodes in their neighbourhood, as we explain later how they obtained this information. The time is measured in slots and each one hop transmission is equal to one time slot.

3.4.1 Protocol Statement

Our communication protocol is based on TSNC introduced by [37]. The level of sparsity is tuned, meaning that at the beginning sparse codes are transmitted and towards the end of the transmissions contain denser codes. Thus, the protocol uses a systematic approach, where first each node sends M uncoded packets and then, the nodes transmit combinations using only the original source packets. The coding process is performed over $\mathbf{GF}(2)$. We use the same parameter C to express the density of the coded packets. A coded packet is C-sparse if it contains C non-zero coefficients from $\mathbf{GF}(2)$.

If less then C source packets are available at a node, then the selection of the source packets is performed among the existing ones. The scheme is different from the original TSNC in the sense that includes also two feedback mechanisms, explained in the following.

- *The explicit feedback* is sent by the BS and flooded to all the nodes. It contains information about seen packets (degrees of freedom) at the BS, where the seen packets have the same meaning as explained in [27]. This feedback is transmitted when at least L new source packets are seen at the BS. The cost for the explicit feedback is equal to the minimum number of hops between the BS and the farthest node from BS in the network.

- *The implicit feedback* appears when the upstream nodes hearing the data packet sent from a downstream node update their buffer state, such that the packets transmitted by the downstream node and found also in the buffer of the upstream node will not be included in future transmissions by the upstream node. Thus, this feedback provides information about the original source packets decoded at the downstream nodes and introduces no cost to the completion time.

Both feedback mechanisms influence the density of the coded packets in the sense that the source packets at a node can be updated by the implicit and/or explicit feedback.

3.4.2 Assumptions

The BS knows the topology of the network, performs all the required algorithms, i.e., finding the downstream and the upstream nodes, the simultaneous possible transmissions, and distributes the information through the network. The nodes are all synchronized and each node knows about its downstream and upstream nodes from the information obtained from the BS. All the nodes have the same transmission range and the erasures are the same for each channel. We also assume that a node can only receive or transmit, but not both at the same time.

3.4.3 Implementation Details

For the detailed implementation, we have the following steps. First, we deploy the network, second, we analyse the topology of the network generated in first step by using graph theoretic analysis, and then, we apply a communication protocol and check the performance. The network deployment is

described in the following, while the metrics used to analyse the topology of the network are given later in the numerical results.

Three steps are required for the network deployment.

- *Generate randomly the nodes*: Each position is generated randomly given the coordinates (x_i, y_i) within the square area and each node has associated an ID from $1, .., N$, where the BS has the ID equal to 1. Given the coordinates of each node, we generate the space matrix S, which defines the distribution of the nodes in the network, such that no node is isolated. Thus, the matrix S is a square matrix whose elements are whether the ID of a node, in the position of the node, or zero otherwise.

- *Find the upstream and the downstream nodes*: Using Algorithm 2, we find the upstream and the downstream nodes. The input data refers to the number of nodes, N, the transmission range defined in meters, R, the coordinates of each node, (x_i, y_i), and the matrix S. The output of the algorithm is given by the upstream and the downstream nodes for each node i, U_i and D_i, respectively. The idea is to count the number of hops away from the BS for each node. If more paths are available, the closest one is chosen. Thus, for each node i, we select the area range and compare the number of hops for each pair of nodes $(i, i' = S(j, k))$, where i' is each of nodes from the area range of i, with $i' \neq i$. The comparison gives us the meaning of node i' for node i, e.g., if the number of hops for node i', called $H(i')$, is higher than the number of hops for node i, called h, then i' is a upstream node for node i.

- *Schedule the transmissions*: Algorithm 3 defines the sets of simultaneous transmissions, given the results from Algorithm 2, U_i and D_i. The algorithm starts by generating the sets with all possible neighbours for each node, then each two nodes can transmit simultaneously if the intersection of their sets is null. From the all possible simultaneous transmissions for a node i, I_i, we further check the unique transmissions over all the network and conclude with the set of simultaneous transmissions, N_i, for each node i.

The transmissions are scheduled in two ways. (1) A Time Division Multiple Access (TDMA), where fixed time slots are allocated to the nodes. A node occupies a whole slot for transmission whether it has or not packets to transmit. The transmission process stops when all the source packets are received at the BS. The scheduled slots are provided by Algorithm 3. (2) The

Algorithm 2 Define the upstream and the downstream nodes for each node

Data: number of nodes (N), transmission range (R), coordinates of each node (x_i, y_i), $\forall i = 1, 2, ..., N$, the matrix S
Result: U_i (upstream), D_i (downstream)
H: array with the number of hops between each node and the BS
h: counter for the number of hops, starting from 0 for the BS
while for each node i **do**
  h is updated for i according to H
  for $j = max(1, x_i - R)$ **to** $min(size(S), x_i + R)$ **do**
    for $k = max(1, y_i - R)$ **to** $min(size(S), x_i + R)$ **do**
      if $(S(j,k) == 0)$ **then**
        continue
      end if
      if $(H(S(j,k)) == h)$ **then**
        Node from $S(j,k)$ is a downstream node for i: $D_i = S(j,k)$
        continue
      end if
      if $(H(S(j,k)) > h)$ **then**
        Node from $S(j,k)$ is a upstream node for i: $U_i = S(j,k)$
        continue
      end if
      if $(H(S(j,k)) == 0)$ **then**
        Increase number of hops for node $S(j,k)$: $U_i = S(j,k)$
        $H(S(j,k)) = h + 1$
        continue
      end if
    end for
  end for
end while

Algorithm 3 Define the set of possible simultaneous transmissions

Data: $U_i, D_i, \forall i = 1, 2, ..., N$
Result: The set of possible transmissions for each node i, N_i
$B_i = U_i \cup D_i$: define the neighbour nodes for each node i
I_i: define all possible simultaneous transmissions for each node i
$I_i = \{\}$
for $i = 1$ **to** N **do**
  for $j = 1$ **to** N **do**
    if $B_i \cap B_j == 0$ **then**
      add j to I_i
    end if
  end for
end for
$N_i = \{\}$
while for each node j from I_i, $i \neq j$ **do**
  if $i \in I_j$ **then**
    add j to N_i
  end if
end while

system uses a random access mechanism, e.g., the nodes for transmission are selected uniformly at random. If a node has no packet to transmit, then the next node is selected. This happens without adding any additional time slots to the total completion time. If two or more nodes in the same transmission range transmit, then a collision occurs. In fact, they collied whenever the areas selected by their transmission range intersect. In this case, we consider that none of the packets coming from the collied nodes is correctly received by any node in the vicinity, but we count the time slot used for transmission. We denote this scheme by using simple the word random.

3.4.4 Comparison Schemes

Two well-known schemes are used for comparison. The description of each one is given in the following.

Master Slave Protocol

The master, in this case the BS, requests from a slave, a node in the network, to transmit its data messages. Each node in the network is aware of its downstream and upstream nodes. Thus, when a node transmits, called the

reference node, the data is saved only by the downstream nodes that hear the transmission. Then, at random, one of the downstream nodes attempts to feedback to the reference node the reception of the data and, in this way, the reference node stops sending any further data. At this point, the downstream node is the new reference node, which broadcasts the data forward. The process continues in the same manner with the other nodes and ends when the data messages from all the nodes reach the BS. The completion time is defined by the summation of all the time slots required for each node to transmit its packets to the BS. We denote this protocol by MS.

RLNC Protocol

This is a scheme where a node combines the packets (source packets and/or other coded packets) with some coefficients selected uniformly at random from a finite field. The results are performed as in the case of TSNC over $\mathbf{GF}(2)$. Nodes know about the downstream nodes and there are no restrictions for the number of packets used in a combination. RLNC does not use the implicit feedback mechanism, but the explicit feedback can be applied.

3.5 Performance Evaluation

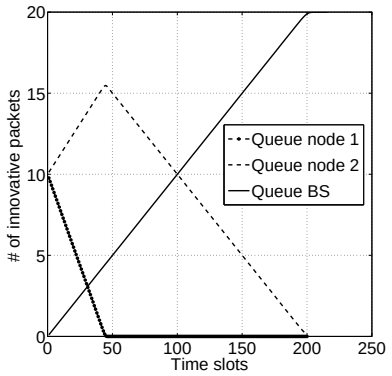
We perform numerical results for the line network, as explained in 3.3, and the grid network, as described in 3.4.

3.5.1 Numerical Results for the Line Network

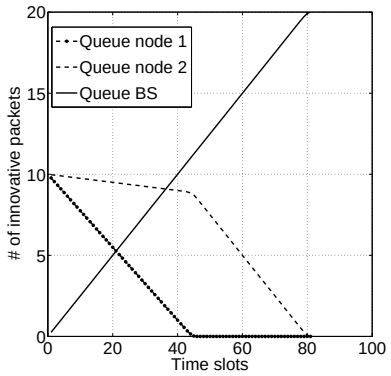
We organise the numerical results for a line network as follows. First, we use the pure RLNC case and show the impact of the K parameter on the completion time. Second, we provide examples with the limitations of the proposed bounds. Third, we give some examples of the completion time using the pure RLNC case, various levels of sparsity, and the feedback techniques.

3.5.1.1 The Influence of the K Parameter

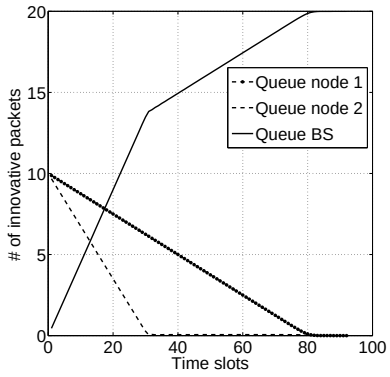
We provide the influence of the K parameter on the completion time for the simplest case, pure RLNC. Figure 3.4 shows an example for the changes in the queue size for the innovative packets and the impact on the completion time at the BS. We consider two nodes, the same transmission probability, and different erasure probabilities. The first case is when the first node has a much better channel than the second, with $K_1 = 0.225$ and $K_2 = 0.1$. In this case, before 50 slots, node 1 finishes its transmissions, while the number



(a) $e_1 = 0.1$, $e_2 = 0.8$



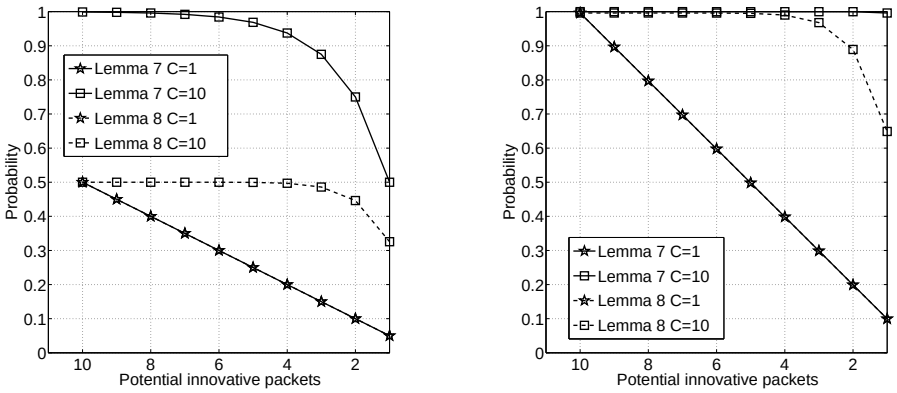
(b) $e_1 = 0.1$, $e_2 = 0.5$



(c) $e_1 = 0.5$, $e_2 = 0.1$

Figure 3.4: Queue size of the innovative packets, for two nodes in a line network, each node transmits 10 packets, field size $\mathbf{GF}(2^8)$, different erasure probabilities, and the transmission probability for each node is 0.5.

of innovative packets at the second node increases because $K_2 < K_1$. In the second case, the node 2 has $e_2 = 0.5$, thus, $K_1 = 0.225$ and $K_2 = 0.25$. Hence, the number of innovative packets at both nodes decreases and the first node finishes faster its transmissions. In the third case, $e_1 > e_2$, with $K_1 = 0.125$ and $K_2 = 0.45$, and thus, the second node finishes its transmissions faster. The completion time is given by the node that takes longer the transmission of the packets, hence, node 2 for the first two cases and node 1 for the third case. This example also confirms equation (3.3).



(a) $q = 2$ (b) $q = 2^8$

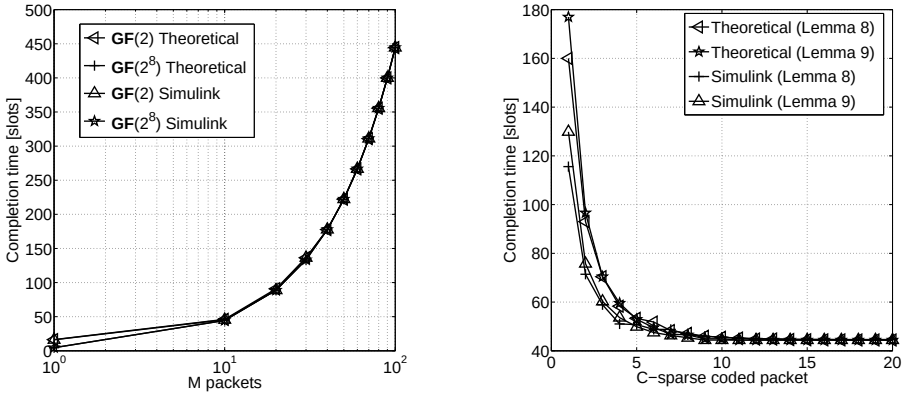
Figure 3.5: The probability that the coded packet is innovative from C selected packets, for $M = 10$, $C = 1$ and $C = 10$, using Lemma 7 and Lemma 8.

3.5.1.2 The Limitations of the Expressions

We determine the limits of our proposed expressions for the sparsity case, ignoring the effect of the erasure and the transmission probability. Thus, we compare the bounds given by the approximated expression in (3.10) and (3.8), for field size $\mathbf{GF}(2)$ and $\mathbf{GF}(2^8)$ and sparsity C=1 and C=10, as Figure 3.5 illustrates. The results show that (i) for $\mathbf{GF}(2)$ and C=1 the expressions coincide, while for C=10 our approximated equation in (3.10) is significantly below the other one, and (ii) for $\mathbf{GF}(2^8)$ and C=1 the expressions coincide and for C=10 they go along for almost of the time. This clearly means that for high field sizes the expression in (3.10) is reliable.

3.5.1.3 Completion Time

Figure 3.6 provides the results for the completion time when varying the number of packets and the sparsity, using the theoretical expressions and the Simulink tool [71]. For the variation of the number of packets, we use pure RLNC case, expression (3.10), and observe that the difference between $\mathbf{GF}(2)$ and $\mathbf{GF}(2^8)$ exists only when a small number of packets is transmitted. This happens because the effect of the field size becomes negligible when the number of packets increases, i.e., q^{-M} from $(1 - q^{-M})$ tends to 0 [57]. Moreover, the expressions from (3.10) and (3.12) support the results for the variation of the sparsity, from $C = 1$ (no coding) up to $C = 20$ (pure RLNC) for both nodes. The completion time decreases significantly after $C = 1$ and tends to 44 time slots.



(a) Varying the number of packets M

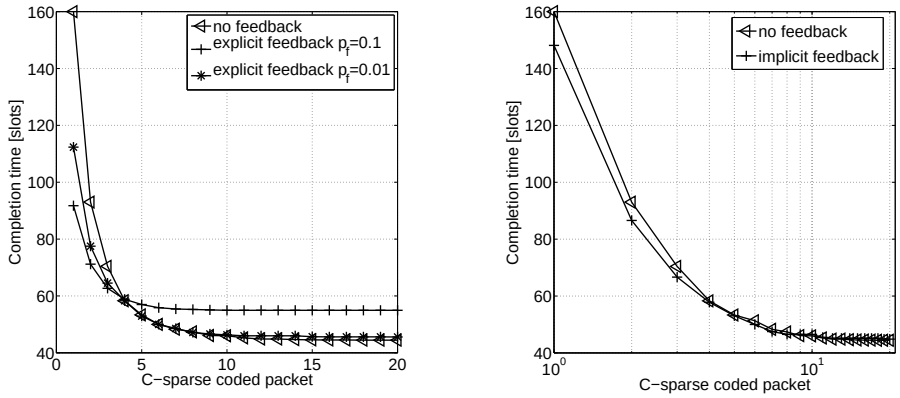
(b) Varying the sparsity of the coded packet

Figure 3.6: Completion time for a line network, $N = 2$, $e_1 = e_2 = 0.1$, $p_1 = p_2 = 0.5$: (a) Packet variation for pure RLNC, $\mathbf{GF}(2)$ and $\mathbf{GF}(2^8)$ and (b) Sparsity variation for $M = 10$ and $\mathbf{GF}(2^8)$.

Then, we perform the mean completion time using feedback, and compare with the results without feedback in Figure 3.7. For the explicit feedback case, the transmission probability for each node is $p_1 = p_2 = 0.5$ and the probability of transmitting the explicit feedback is $p_f = 0.01$ and $p_f = 0.1$, respectively. This means that 2% and 20%, respectively, of the total time of the system is allocated for the feedback transmission. Thus, for a more often feedback, up to $C = 4$ the completion time decreases, but then, it tends to 55 slots, which means a higher completion time than in the case without feedback. This happens because the explicit feedback introduces costs for its transmission. For a less frequent feedback, up to $C = 4$ the completion time is reduced, but not so much as in the case of $p_f = 0.1$, and after $C = 4$, the completion time for the feedback case becomes close to the one without feedback. In case of implicit feedback, the findings show that for only two nodes, the implicit feedback can slightly improve the completion time when the codes are very sparse. From these two examples, we conclude that both the explicit and the implicit feedback are useful when applied for sparse codes.

3.5.2 Numerical Results for the Grid Network

We divide the numerical results for the grid network into: (a) the transmission range variation, where we use two examples and different metrics to characterize the network topology, and (b) the sparsity, the feedback, and



(a) Varying the sparsity of the coded packet with explicit feedback

(b) Varying the sparsity of the coded packet with implicit feedback

Figure 3.7: Completion time for a line network varying the sparsity, $N = 2$, $e_1 = e_2 = 0.1$, $p_1 = p_2 = 0.5$, $M = 10$ and $\mathbf{GF}(2^8)$, for the effect of (a) explicit feedback, $p_f = 0.01$, $p_f = 0.1$ and (b) implicit feedback.

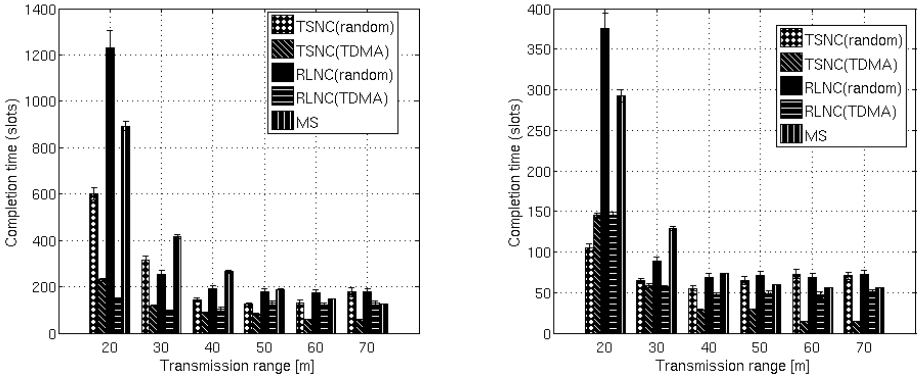
the number of packets variation, where the transmission range is fixed. The results are given for 100 repetitions of each experiment and using 97.5% confidence intervals.

3.5.2.1 Transmission Range Variation

We use the following metrics to describe the network topology:

- *Node degree* is the number of branches a node connects to.
- *Path length* is average shortest path between any pair of nodes.
- *Algebraic connectivity* shows how well a network is connected and how fast information data can be shared across the network. If it is higher than 0, the network is a connected graph, while if it is close to 0, the network is close to being disconnected.
- *Clustering coefficient* is the ratio of the nodes tending to cluster together.

We vary the transmission range for various topologies and for each topology we compute the mentioned metrics. The results are given in Table 3.1, where the metric's variation is shown only for the minimum and the maximum value of the range, 20 m and 70 m, respectively. The examples are for 30-node topology and 15-node topology. The metrics show that the networks



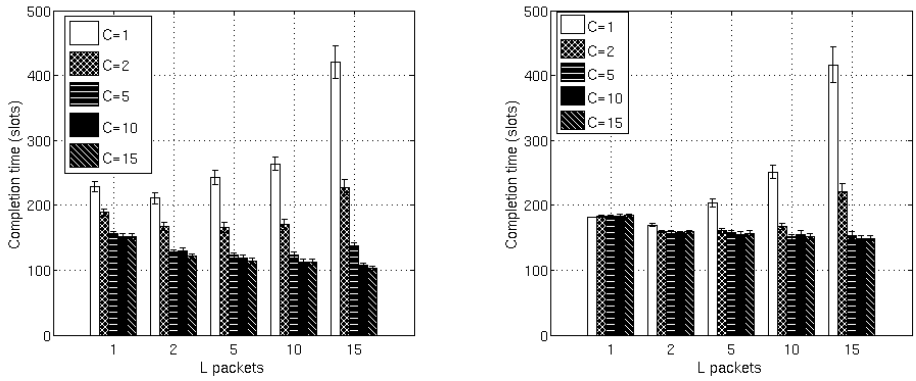
(a) 30-node case (b) 15-node case

Figure 3.8: Completion time when the range is varied, $M = 1$, erasure probability is the same for all the channels and equals 0, $C = L = 50$.

become better connected and tend to a cluster when the range is increased. The results in terms of completion time are given in Figure 3.8 for the variation of the transmission range. Here, the TSNC protocol uses dense codes (pure RLNC) and the explicit feedback is sent only at the end of the transmission process ($C = L = 50$). In this example, the difference between TSNC and RLNC is given by the fact that the first one uses the systematic approach for sending the packets, uncoded packets first and then coded packets, and the implicit feedback mechanism. The worst protocol is RLNC using random scheduling or MS depending on the transmission range. When the range increases to 70 m, according to Table 3.1 the nodes become almost all directly connected to the BS. In this case, the protocols using random scheduling offer the worst performance, because allocating time slots using TDMA becomes simpler for this case, e.g., the nodes transmit in order, each node at the time. Thus, we conclude that for this type of network (a) TSNC using TDMA outperforms the other protocols when the nodes become directly connected to the BS and (b) the choice of the scheduling mechanism appears to be favourable for TDMA for the most situations.

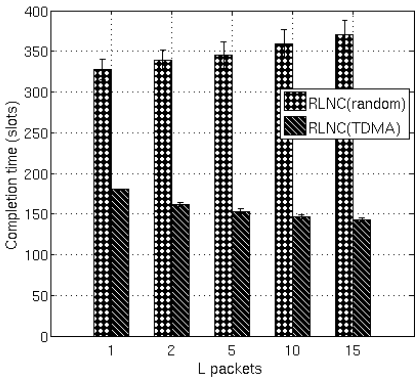
3.5.2.2 Protocol Performance for Fixed Transmission Range

We choose 15-node case, fix the range to 20 m, and vary the sparsity, the frequency of the explicit feedback, and the number of packets, as Figure 3.9 and Figure 3.10 show. In Figure 3.8, 15-node case for range 20 m, TSNC with random scheduling performs better than TSNC with TDMA, while RLNC with random scheduling worst than RLNC with TDMA. Here, we want to



(a) TSNC (random)

(b) TSNC (TDMA)



(c) RLNC

Figure 3.9: Completion time for 15-node case, transmission range R = 20 m, for $M = 1$, the erasure probability is the same for all the channels and equals 0, when varying the sparsity and the explicit feedback for (a) TSNC (random) and (b) TSNC (TDMA), and when varying the explicit feedback for (c) RLNC schemes, $C = 15$.

Table 3.1: Topological characteristics, area=100 m^2, range varies between 20 m and 70 m.

Case	Links	Node degree	Path length connectivity	Algebraic connectivity	Clustering coefficient
30-node	106	[3.7,..,24.8]	[3.9,..,1.1]	[0.1,..,15.2]	[0.5,..,0.9]
15-node	15	[4,..,13.6]	[2.4,..,1.02]	[0.2,..,11]	[0.6,..,0.97]

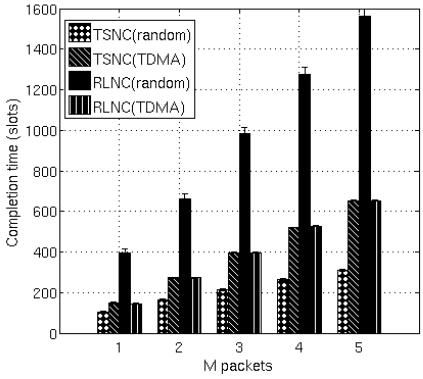


Figure 3.10: Comparison of the completion time for various schemes for 15-node case, transmission range R = 20 m, when varying the number of packets, $C = L = 50$, the erasure probability is the same for all the channels and equals 0.

understand if by varying the sparsity, the explicit feedback, and the number of packets, these protocols maintain the same position. We classify the results as in the following, the sparsity only for TSNC and the explicit feedback for both TSNC and RLNC.

(a) *Sparse network coding case* : The findings show that for $C = 1$ the completion time increases significantly comparing to the other cases. Also, when the sparsity decreases, the completion time converges to a constant value. For instance, the difference between $C = 5$ and $C = 15$ is irrelevant for the completion time. But, the encoding and decoding complexity are significantly influenced, e.g., higher is C more operations are required for performing encoding and decoding operations, as well as the more overhead is introduced by the coefficients sent together with the coded packets.

(b) *Explicit feedback case* :

$TSNC$: The minimum completion time can be achieved for different values of L. Thus, the worst performance is achieved when $C = 1$ and $L = 15$. We also observe that the density of the sparse packet is highly influenced by the explicit feedback. Thus, the less frequent is the explicit feedback, the higher is the difference between very sparse coded packets and denser coded packets, for both cases with random scheduling and TDMA. For instance, in the case of TSNC with random scheduling if $L = 1$, the completion time for $C = 1$ is 1.5 times higher than for $C = 15$, while if $L = 15$, the completion time for the same case becomes 4 times higher. We conclude that for a grid network, the minimum completion time is achieved for very sparse coded packets by using a frequent feedback and for dense coded packets by using a less frequent feedback.

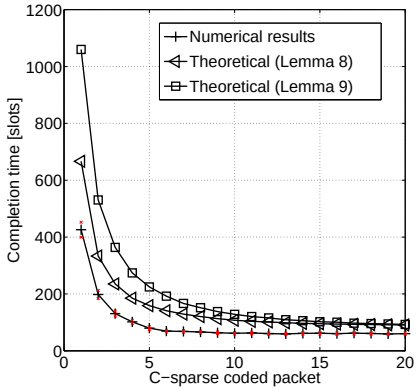


Figure 3.11: Completion time for a line network varying the sparsity, $N = 2$, $e_1 = e_2 = 0.1$, $p_1 = p_2 = 0.5$, $M = 10$ and $\mathbf{GF}(2)$, comparison between numerical and theoretical results.

RLNC: In case of TDMA, we get a decrease when L increases and for random scheduling, we observe a smooth increase of the completion time when L grows.

In conclusion, for this case, TSNC with random scheduling outperforms TSNC with TDMA for specific values of C and L, especially when the sparsity decreases and the explicit feedback becomes less frequent. We also observe that when $C = 1$ and frequent explicit feedback, TSNC with TDMA is a better choice. For RLNC, for any value of the explicit feedback, the random scheduling maintains worst performance than TDMA.

(c) *Number of packets case* : In Figure 3.10, the completion time grows differently for the presented protocols by increasing the number of packets. Thus, for this scenario, TSNC with random scheduling outperforms the other protocols, while the RLNC with random scheduling gives the worst performance.

In conclusion, we observe that the sparsity and the explicit feedback depend one on each other and all the other factors, such as the topology, scheduling mechanisms, and the protocols, contribute to the gains of the system.

3.5.3 Comparison Numerical Results vs Theoretical Results

The results for the mathematical model are divided in: the pure RLNC case, the sparse network coding case with no feedback, explicit feedback and implicit feedback with sparse network coding case. The pure RLNC case

is the classical RLNC, where all the existing packets in the node's queue are combined, including the original source packets and the coded packets. The sparsity with no feedback and sparsity using feedback techniques are tuned versions of RLNC, because (a) in the sparsity with no feedback case, the sparsity level is fixed during all the transmission process, (b) both, the implicit feedback and the explicit feedback, are optionally used, and (c) the explicit feedback also introduces delays, but it is different than the numerical results because the downstream node sends to the upstream node information about all the non-innovative packets of the upstream node.

For the numerical results given by the grid network, the TSNC protocol uses a systematic approach for the transmission of the packets, meaning uncoded packets first and then coded packets. The coding is performed using only the original source packets. The level of sparsity is also changed by the implicit feedback, which is always used, and by the explicit feedback, which is optionally used and introduces costs. Moreover, the explicit feedback provides information only about specific packets. The RLNC protocol does not use the implicit feedback or various sparsity levels for the coded packets, but it can include the explicit feedback.

In terms of the limitations, the theoretical results are valid for a line network, any number of nodes, number of data packets, and field size, while the numerical results are valid for an arbitrary network topology, any number of nodes and number of data packets, but only for $\mathbf{GF}(2)$.

In the following, we compare the numerical results, using 97.5% confidence intervals, with the results from the mathematical model, using the bounds given by the expressions (3.10) and (3.12). We take a simple example for two nodes in a line network, $\mathbf{GF}(2)$, no feedback mechanism, while varying the sparsity of the coded packets. Given that we choose the C parameter and then select the coefficients at random using a field size $\mathbf{GF}(2)$, meaning that the coefficients are generated independently using a random variable with probability of success 0.5, the "effective C" is really C/2. Similar reasoning we use for the density case, equation (3.12). The findings are given in Figure 3.11. The results show that the theoretical results represent upper bounds and the expression (3.10) is closer to the numerical results.

3.6 Concluding Remarks

Inspired by the previous work in [36], we proposed to understand the feasibility of using TSNC and RLNC in more general scenarios. Thus, we addressed the design for this class of protocols by using various sparsity levels of the coded packets and different feedback types. We measure the performance

of the network using as metric the completion time, the total time required to gather the data packets from the nodes at a BS. First, we proposed a mathematical model using the differential equations to describe the communication process in a line network. Here, our expressions are valid for any field size and represent approximations for (a) the pure RLNC case, (b) the sparse network coding case, (c) the sparsity with explicit feedback case, and (d) the sparsity with implicit feedback case. The results show that the completion time is minimum when the codes are denser. We conclude that both implicit and explicit feedback are useful in terms of completion time, but only when applied for sparse codes. Second, we performed numerical results for a grid network using a small field size and compared the findings with a master slave protocol, conventional RLNC and with some tuned versions of RLNC. The results show that the network topology dictates the protocol performance, where TSNC using TDMA outperforms other protocols when range is increased. Moreover, the completion time becomes worst when the sparsity is increased. We also observed that the sparsity of a coded packet is influenced by the explicit feedback in the sense that the less frequent the explicit feedback, the higher the difference between very sparse coded packets and dense coded packets. We conclude that for a grid network, the minimum completion time is achieved by using very sparse coded packets and frequent feedback or dense coded packets and less frequent feedback.

Chapter 4

Hardware Abstraction Model

4.1 Motivation

Chapter 2 and Chapter 3 analysed the data transmission time using as metrics the delay and the completion time. These metrics are fundamental for the energy efficiency of the system, because the energy consumption is derived from the transmission time. Hence, in this chapter, given the disruptive nature and potential benefits of network coding to enhance network performance [5, 8, 72], we set out to understand network coding feasibility in terms of energy consumption. Although network coding is known to reduce the number of transmitted packets on the one side, particularly in multiple receiver/transmitter scenarios [13], on the other side it requires also additional processing. This may be significant for the battery-powered networks because of the network coding impact on the total energy efficiency over the network.

Conventional wisdom in communications engineering says that transmission power is the most important factor in determining the energy consumption of data gathering schemes for battery-powered networks, e.g., WSN. It is therefore not surprising that protocol design for this class of networks has been mostly concerned with reducing the number of transmitted bits, while ignoring other sources of energy consumption, such as sleeping, data processing, receiving, or switching between sending and receiving states. However, sensor networks are heterogeneous in the sense that the underlying hardware of each sensor may vary considerably. Although the transmission power does dominate the energy budget for some sensors [73], [74], [75], it has become apparent that this is not true for many commercially available sensor platforms. A closer look at the technical specifications of common devices reveals the importance of several other sources of power consumption that affect the

total energy consumption of the system. As an example, it is not difficult to find instances in which the receive power can be higher than the transmit power [51], [76], [52]. The ratio might vary in the order of tens and even hundreds. Another component that is usually ignored is the power required for computation. Some of the sensors feature processing power equal to half the transmit power or more [77], [78], [79]. Often, idle/listen power is almost equal to the receive power, [52], [80], [81], which increases the total energy consumption even more. Also, the switching between different modes can take non-negligible time and energy [46].

It is also worth mentioning that sensor nodes can operate under various power consumption modes. The most power consuming one is the active mode, when both the radio and the μC are active and the sensor can send or receive data. The idle\listen mode is when the sensor is neither sending nor receiving any data yet can listen to the channel. Finally, several types of sleep modes are possible depending on the components that are turned off. The power consumption can be very low at times, depending on how the sensor node was programmed to meet the requirements of the application

The processing time is highly dependent on the operations performed at the sensor, and a deeper understanding on the implementation costs of network coding in different sensors is needed to the successful design of coding-aware protocols. For example, the microcontroller (μC) of the sensor node is usually built as a simple device with the power consumption approximately constant. Thus, when using network coding at the sensor node the time for generating the coded packets will have a significant influence on the energy consumption profile of the protocol.

We conclude that the transmit, receive, process, idle\listen, switching and sleeping modes cannot be ignored, when designing protocols for sensor networks. This calls for a hardware abstraction that addresses the overall energy consumption in sensor nodes and allows for fast evaluation of the energy efficiency of the given protocol specifications. Our main contributions are:
(a) *Total Energy Model*: We define the total energy model based on the transmit, receive, processing, idle\listen, sleeping and switching energy. We also identify the main energy parameters and show that existing sensor technologies can be divided in several classes with specific energy characteristics.
(b) *Communication Strategies*: We apply the model to a star topology network and illustrate the energy consumption of two different protocols, including (1) TARQ: Time Division Multiple Access (TDMA) with Automatic Repeat reQuest (ARQ), and (2) TNC: TDMA with network coding, under two different platforms (i.e., different commercial sensors). Network coding constitutes an instance of computation at a sensor μC that is shown to have

a non-negligible cost in energy.
(c) *Experimental Results*: We implement network coding on TelosB motes and identify the main obstacles caused by the μC processing.
(d) *Hardware Dependency*: We show that the choice of communication strategy depends on the hardware characteristics.

The remainder of the chapter is organized as follows. Section 4.2 provides an overview of the related work. The description of the total energy model is given in Section 4.3, with the identification of the main consumption parameters, the proposed hardware abstraction and the definition of the communication protocols. The numerical results are given in Section 4.4, including the real-life measurements. The chapter concludes with Section 4.5.

4.2 Related Work

Minimizing the communication cost of sensors has constituted a key concern for ensuring low energy consumption in WSN. Energy efficiency of communication protocols has been analyzed using the sensor's radio characteristics in [46], [82], [83] and [47]. In fact, [82] provided optimization mechanisms for parameters such as data rate, power consumption in transmit, receive, sleep and switch mode. [46] characterizes the switching energy for transitioning between two modes of operation, while [47] analyzes the benefits and caveats of deep sleep and light sleep modes. Studies that analyze the interplays amongst energy consumption, switching energy between different modes, and the battery capacity have also drawn some interest [83]. A different approach to understanding energy efficiency in sensors has focused on the computational energy of the sensor. In particular, [47] analyzed the cases when both the μC and radio are waked-up for transmission/sensing, but not for extra processing, e.g., encryption or coding of data packets. None of these works consider a hardware abstraction of the sensor nodes, including a complete characterization of the radio and μC, tremendous significant for the total energy consumption of the WSN.

In terms of encoding complexity for network coding, [84] proposes a scheme with $O(L)$ complexity at the source, where L is the number of bits in each transmitted packet for multicast sessions. Network coding has also the potential to minimizing the time and/or energy to transmit a batch of data packets for half-duplex channels by providing protocols that adapt to channel, traffic, and transmission energy profiles [85]. Our work was inspired in part by the observations of Shi *et al.* [40], which present guidelines for protocols in body area networks that take into consideration the transmit

power, the receive power and the processing power of the sensing devices.

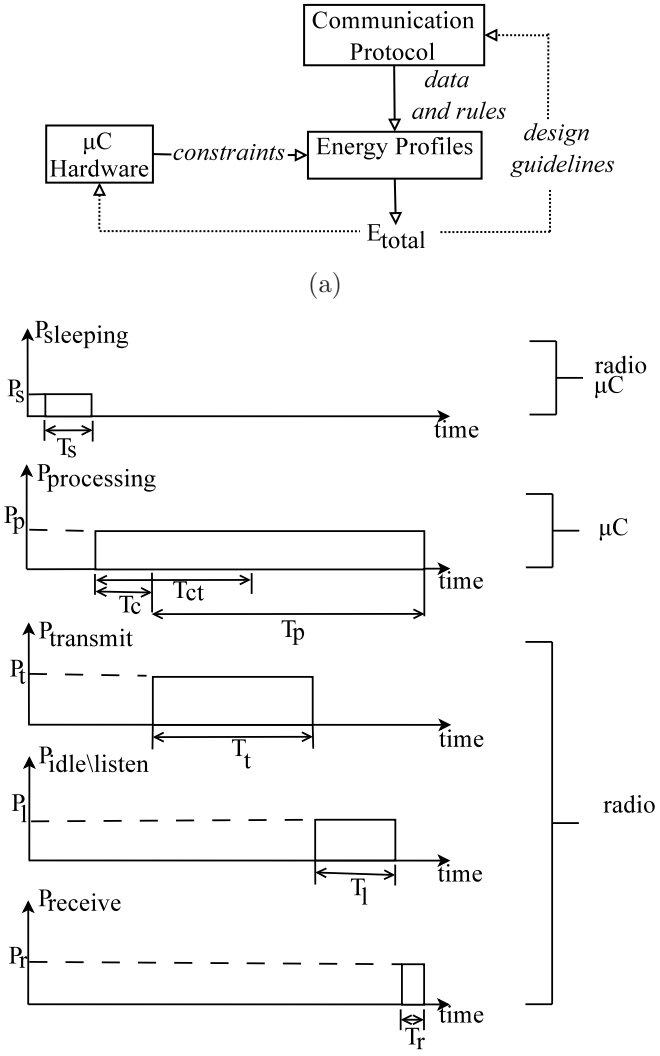


Figure 4.1: (a) Total Energy Model Description, (b) Time diagrams for one node showing sleeping, processing, transmit, idle\listen, and receive power.

4.3 Total Energy Model

The sources of energy consumption in a sensor are directly linked to different hardware components. Although this work considers two main elements, namely the μC and the radio, additional hardware components could be mod-

elled and brought into our framework on a sensor-by-sensor basis. However, the interactions among these sources of energy consumption, the sensor's functionalities, and the communication protocols are more relevant to characterize the WSN's energy footprint as they provide a holistic view of the problem. The total energy model is thus described as three main blocks (see Figure 4.1(a)): (a) the energy profiles available at the sensor, (b) the hardware characteristics of the μC that give the constraints for the energy cost, and (c) the communication protocol that defines the data exchange process. The blocks are explained in the following.

4.3.1 Energy Profiles

We describe in detail the different energy profiles available at a sensor node and use Figure 4.1(b) to describe some of these profiles and their interactions. The parameters used in Figure 4.1(b) illustrate the active periods of different energy profiles, and their corresponding hardware component for the case of a sensor waking up from sleep mode to transmit a data packet and receive a (control or data) packet.

Our energy model incorporates the power required by each profile as well as their active periods. Since the power level is considered fixed per profile, the main challenge is to characterize the active time of the various energy profiles in order to determine their energy consumption.

4.3.1.1 Transmit Energy

It refers to the energy consumed when the radio of the sensor is in transmit mode and sends packets to the network. The transmit energy is given by $E_t = T_t P_t$, where T_t and P_t correspond to the transmission time and power. P_t may vary, as shown in Table 4.1.

4.3.1.2 Receive Energy

It is the energy consumed when the radio is in receive mode and the sensor receives data from another node. By looking at different sensor models, it is clear that receive power can be either smaller or greater than the transmit power. Defining $\alpha = P_r/P_t$ as the ratio between transmit and receive powers (P_r), Table 4.1 illustrates that α varies by a factor of 10 and even 100 depending on the sensor, with $P_r \geq P_t$ in several cases. Thus, the expression of receive energy is given by $E_r = T_r P_r = \alpha T_r P_t$, where T_r constitutes the receive time.

Table 4.1: Transmit power P_t, receive power P_r, and the ratio α for different sensor models.

Sensor Model (radio)	$\mathbf{P_t}$**[mW]**	$\mathbf{P_r}$**[mW]**	$\boldsymbol{\alpha}$
Rene (TR1000) [87]	48	15.2	0.32
TelosB (CC2420) [76], [88]	[25.5 52.2]	56.4	[1.08 2.21]
TelosB CM5000 (CC2420) [89]	[25.5 52.2]	56.4	[1.08 2.21]
Tmote Sky (CC2420) [90]	[30.6 63]	70.92	[1.13 2.31]
Imote2 (CC2420) [88], [91]	[30.6 63]	67.68	[1.13 2.31]
Lotus (802.15.4) [77]	[33 56.1]	52.8	[0.94 1.6]
Mica2 (868/916) [92]	89.1	33	0.37
Mica (TR1000) [93]	36	5.4	0.15
MicaZ (802.15.4) [51]	[36.3 57.42]	65.01	[1.13 1.79]
IRIS (802.15.4) [94]	[33 56.1]	52.8	[0.94 1.6]
Mulle (AT86RF230) [95]	[32.3 56.1]	52.7	[0.94 1.63]
Eyes (TR1001) [80]	36	11.4	0.32
Bean (CC1000) [81]	[15.9 80.01]	28.8	[0.36 1.81]
BTnode (CC1000) [81]	[15.9 80.01]	28.8	[0.36 1.81]
Zolertia Z1 (CC2420) [79]	[25.5 52.2]	56.4	[1.08 2.21]
FireFly (CC2420) [96]	[25.5 52.2]	56.4	[1.08 2.21]
Ubimote2 (CC2520) [97]	[56.76 73.92]	40.7	[0.55 0.71]
Waspmote 1 (XBee 802.15.4) [52]	148.68	150.78	1.01
Waspmote 2 (XBee PRO 56) [52]	562.74	171.24	0.3
Waspmote 3 (XBee ZigBee) [52]	113.94	113.04	0.99
Waspmote 4 (XBee ZB PRO 45) [52]	315	151.38	0.48
Waspmote 5 (XBee 868) [52]	405	219	0.54
Waspmote 6 (XBee 900) [52]	231	198	0.86
Waspmote 7 (XBee XSC) [52]	145.05	330	2.28
VEmesh-LP (SX1211/1231) [98]	108	108	1
VEmesh-HP (SX1211/1231) [98]	2900	108	0.04
Tinynode 584 (XE1205) [99]	[122.4 255]	75.6	[0.3 0.61]
Tinynode 184 (SX1211) [100]	[75.6 90]	12.6	[0.14 0.16]

4.3.1.3 Processing Energy

It is the energy consumed by the μC while it a) receives data from other sensors, b) collects and processes data, and c) executes the algorithms and decisions required by its communication protocols. According with the hardware models from Table 4.2, the power consumption when the μC is active can achieve high values comparing with the transmit power, i.e., $P_p \geq P_t$ in some scenarios. We define the ratio $\beta = P_p/P_t$.

The sensors are typically simple devices in their energy profile, i.e., do not use the sophisticated chip power management strategies, and the μC of the sensor is also built as a elementary component. Thus, the power consumption is approximately constant while executing instructions and this happens because all the μC's components, such as processor core, memory, ADC, oscillator, timer and the input/output peripherals are turned on while it is in active mode. Thus, to compute the processing energy it is sufficient to find the time the μC is active. The time to perform additional operations, e.g., for processing information or perform encryption algorithms, is denoted by T_{ct}, as in Figure 4.1(b). T_{ct} can start before the transmission and last during the sending of some initial data, depending on the requirements of the application and performance of the μC. Thus, we denote by T_c the time to perform computation before the transmission starts. Two extreme cases can be considered for the total processing energy: (a) when all additional processing is performed before the transmission starts, $T_c = T_{ct}$, and (b) when the additional processing is carried out only during the transmission process, $T_c = 0$. Clearly, the μC is active during the transmission, reception and listen modes, as illustrated in Figure 4.1(b). Thus, the energy consumption for processing is given by $E_p = (T_p + T_c)P_p = \beta(T_t + T_r + T_l + T_c)P_t$, where P_p and T_p are the processing power and time and T_l is the listen time.

4.3.1.4 Idle\Listen Energy

It is the energy when the radio of the sensor is active, but neither sending nor receiving any data. For many sensor models the power in idle mode is almost equal to the one in receive mode. We define the energy expression by $E_l = T_l P_l = \gamma T_l P_r = \gamma\alpha T_l P_t$, $\forall\gamma = 0 \ldots 1$, where P_l is the power consumption in idle mode.

4.3.1.5 Sleeping Energy

It refers to the energy when the radio and μC of the sensor are in sleep mode and can be described by $E_s = T_s P_s = \epsilon T_s P_t$, where T_s and P_s are the time

Table 4.2: Processing power P_p and the ratio β for different sensor models.

Sensor Model (μ**C**)	$\mathbf{P_p}$**[mW]**	$\boldsymbol{\beta}$
Rene (ATmega 8535) [101]	16	0.34
TelosB (MSP430F1611) [102]	5.4	0.1
TelosB CM5000 (MSP430F1611) [102]	1.8	0.03
Tmote Sky (MSP430F1611) [90], [102]	6.48	0.1
Imote2 (Marvell PXA271) [91]	111	1.77
Lotus (ARM7 Cortex M3) [77]	165	2.93
Mica2 (ATmega 128L) [92]	26.4	0.30
Mica (ATmega 103L) [93]	16.5	0.46
MicaZ (ATmega 128L) [51]	26.4	0.46
IRIS (ATmega 1281) [94]	26.4	0.46
Mulle (Renesas M16C/62P) [78]	25.84	0.52
Eyes (MSP430F149) [103]	1.68	0.05
Bean (MSP430F169) [103]	1.8	0.02
BTnode (ATmega 128L) [104]	23	0.29
Zolertia Z1(MSP430F2617) [79]	30	0.52
FireFly (ATmega 1281) [96]	18	0.34
Ubimote2 (MSP430F2618) [105]	1.8	0.02
Waspmote 1 (ATmega 1281) [52]	30	0.2
Waspmote 2 (ATmega 1281) [52]	30	0.05
Waspmote 3 (ATmega 1281) [52]	30	0.26
Waspmote 4 (ATmega 1281) [52]	30	0.1
Waspmote 5 (ATmega 1281) [52]	30	0.07
Waspmote 6 (ATmega 1281) [52]	30	0.13
Waspmote 7 (ATmega 1281) [52]	30	0.21
VEmesh-LP (VE209) [98]	0.072	0.0007
VEmesh-HP (VE209) [98]	0.072	≈ 0
Tinynode 584 (MSP430F1611) [99]	9.35	0.04
TinyNode 184 (MSP430F2417) [100]	16.56	0.18

and power consumption in sleep mode and $\epsilon = P_s/P_t$. For the sensors from Table 4.1, ϵ has values less than 0.01.

4.3.1.6 Switching Energy

It is used when the sensor switches from active mode to either idle mode or a sleep mode, or viceversa. The energy consumption in the sleep mode is very low, but the transitions between modes take both time and energy. The decision of a node to switch to a sleep mode is mostly dependent on the hardware model. According to [106], a transition to a low power mode is worthwhile if the time necessary to stay in an idle mode is sufficient large, namely, if $T_l \geq \frac{1}{2}[T_{ls} + \frac{P_l+P_s}{P_l-P_s}T_{sl}]$, where T_{ls} and T_{sl} refer to the time to switch from the idle mode to sleep mode and vice versa.

Finally, we characterize the total energy consumption (E_{total}) for a WSN with N sensors, where we abstract the various hardware energy characteristics as 5 key fixed parameters (α, β, γ, ϵ, P_t) and the active times of each profile, thus,

$$E_{total} = \sum_{j=1}^{N}[E_t^j + E_r^j + E_p^j + E_l^j + E_s^j] \tag{4.1}$$

$$= \sum_{j=1}^{N}[(1+\beta^j)T_t^j + (\alpha^j + \beta^j)T_r^j + \beta^j T_c^j + (\beta^j + \gamma^j\alpha^j)T_l^j + \epsilon^j T_s^j]P_t^j.$$

The latter is determined by the sensor model and the communication protocol implemented. This result illustrates that E_{total} for a communication protocol shall vary significantly from sensor to sensor or, similarly, that the total energy for a WSN depends greatly on the used protocol. We distinguish two cases for E_{total}: (1) for hardware models where we have no control of P_t, and (2) for models where configurable transmission powers are available. For the latter, there is typically a control mechanism that allows to control P_t based on signal strength metrics from the receiver, e.g., Received Signal Strength Indicator (RSSI).

4.3.2 μC's Hardware Abstraction

As previously defined, the radio and μC of the sensor are the main components that contribute to the total energy model. However, when additional cost is added to the sensor, specific μCs may influence differently the energy consumption. In this sense, we show the main steps of the μC's process-

Table 4.3: Cost for coding operations over $\mathbf{GF}(2^m)$.

Operation	**Cost**
1 $\mathbf{GF}(2^m)$ *addition*	m XOR
1 $\mathbf{GF}(2^m)$ *multiplication*	m^2 AND and $1.5(m-1)^2$ XOR
1 $\mathbf{GF}(2^m)$ *register*	m `register write`

ing that count the total time for performing arithmetic operations, namely T_{ct}. In fact, the amount of energy consumed by a sensor to perform a given function is inherently affected by the μCs architecture and capabilities. As a relevant example, we illustrate the key steps at the μC that count towards the total time (T_{ct}) required for completing basic arithmetic operations in Galois Fields, which are essential to implement RLNC. In RLNC, the encoder mixes the packets $p_1, p_2, .., p_M$,each of L bytes, and outputs coded packets of the form $\sum_{j=1}^{M} \varphi_j \cdot p_j$. The coding coefficients $\varphi_1, ..., \varphi_M$ are independently and randomly selected from $\mathbf{GF}(2^m)$. Coded packets are transmitted together with the coefficients used in combinations. At the receiver side, after collecting M linearly independent combinations and by using Gaussian elimination, the original packets can be recovered. In the following, we consider two architectures for data processing and their effect over T_{ct}, although alternative implementations are possible, e.g., using look up tables.

4.3.2.1 Dedicated/Configurable Computing Model

This model is used when a dedicated hardware, e.g., ASIC, or a reconfigurable device, e.g., FPGA, is available. Performing RLNC operations in this case requires the following considerations.
(a) *Cost of operations in* $\mathbf{GF}(2^m)$: is shown in Table 4.3 [107].
(b) *Total number of operations*: for one coded packet is given by $(M-1)$ additions and M multiplications, resulting $[(m+1.5(m-1)^2)M - m]$ XOR and (m^2M) AND logic gates. Each packet has L bytes given the final number of required logic operations as being $\frac{8L}{m}[(m+1.5(m-1)^2)M - m]$ XOR and $(8LmM)$ AND. Furthermore, each coded packet of L bytes contains in the header also the coefficients used in performing the combinations, thus additional $(mM + 8L)$ operations are required for saving the data in the memory.
(c) μC *features*:

- *B-bit model*: gives the final XOR and memory operations, which are divided by B and the AND operations that are divided by (a) m, for

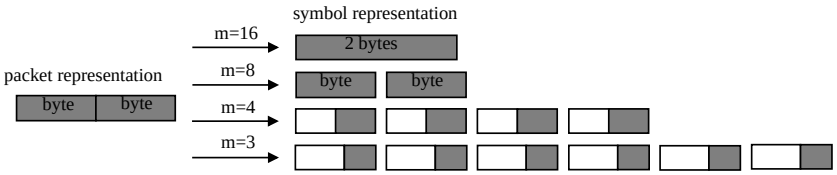


Figure 4.2: Packet to symbol representation at the μC for $L=2$ bytes for an 8-bit microcontroller.

$m \leq B$, and (b) B, for $m > B$.

- *Number of clock cycles*: that one instruction can take denoted by C contributes also to the final processing time.
- *Instruction cycle time*: is well-documented in the description of the μC and we denote it by T_{ict}.

The total time the μC takes to process R coded packets based on all these steps is

$$T_{ct} = \begin{cases} RT_{ict}C[\frac{1}{B}[(1+1.5\frac{(m-1)^2}{m})8L+m]+8L]M, \text{if } m \leq B \\ RT_{ict}C\frac{1}{B}[[(1+1.5\frac{(m-1)^2}{m})+m]8L+m]M, \text{if } m > B. \end{cases} \quad (4.2)$$

4.3.2.2 Software Description Model

This model is based on performing operations using instructions available at the μC. The software implementation of RLNC takes into consideration the field size (m), number of coded packets (R), number of packets (M) and the packet size (L bytes), and is affected by various effects including the μCs data register bit size. Each packet is represented by $\lceil \frac{8L}{m} \rceil$ symbols, which together with the coefficients are used to perform RLNC operations. We identify two possible cases. (i) For $m \in \{2^{n+3} : n \in \mathbb{N}\}$, the conversion of the packets into symbols is straightforward as they coincide with the size of the μCs data registers, e.g., the first two cases from Figure 4.2, where the coloured blocks represent the data. The total coding time includes the time to perform linear combinations using multiplication and addition operations, denoted by T_{lc}, and the time to generate the coefficients, defined by T_{cf}. The addition and multiplication are based on simple XOR operations and the summation of the powers, from the polynomial representation of elements over $\mathbf{GF}(2^m)$, modulo 2^m-1, respectively. (ii) For $m \in \{x : x \in \mathbb{N}^* \setminus \{1, 2^{n+3}\}, n \in \mathbb{N}\}$, two additional conversion functions are necessary (e.g., the cases for $m=4$ and

$m = 3$ in Figure 4.2). One is to convert the each packet into $\lceil \frac{8L}{m} \rceil$ symbols and the other is to convert back the combinations obtained from RLNC operations into packets, namely, coded packets. Both conversion functions are implemented by using SHIFT and AND operations and we denote the time to perform them by T_{cv}. A special case is for $\mathbf{GF}(2)$ where the packets are combined by using simple XOR operations and the coding time is denoted by T_{xor}. Thus, T_{ct} is given by

$$T_{ct} = \begin{cases} T_{cf} + T_{lc}, \text{if } m \in \{2^{n+3} : n \in \mathbb{N}\}, \\ T_{cf} + T_{lc} + T_{cv}, \text{if } m \in \{x : x \in \mathbb{N}^* \setminus \{1, 2^{n+3}\}, n \in \mathbb{N}\}, \\ T_{xor}, \text{if } m = 1. \end{cases} \tag{4.3}$$

4.3.3 Communication Protocols

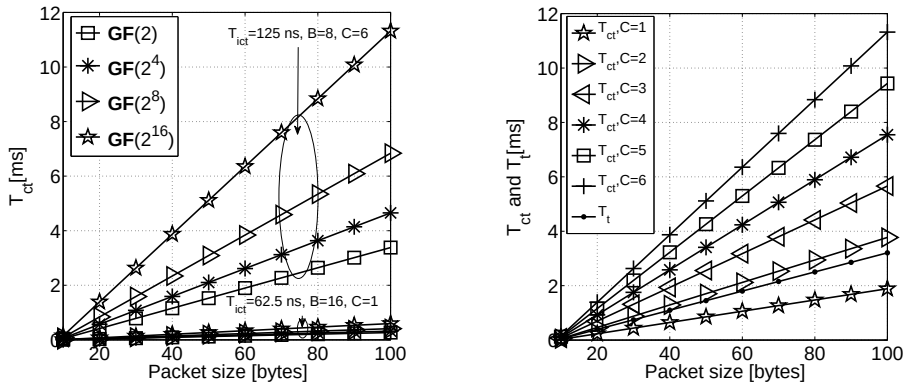
We consider a N-node star topology network, where a sensor node, denoted by S_i, $\forall i = 1, 2 \ldots N$, wants to send directly M packets, each one of L bytes, to the base station (BS). The channels corresponding to each sensor are expressed as erasure channels with erasure probability e_i. Each packet is sent in a time slot and BS acknowledges the nodes about the reception of the packets. Each acknowledgement packet (Ack) has A bytes and is successfully received by all nodes.

We analyze two communication protocols: a coding-aware protocol that requires energy to process each coded packet, and a simple protocol that needs no additional processing for packets. The following settings are valid for both protocols. At the beginning, the BS broadcasts a beacon message to synchronize the nodes. Active nodes send a short length message to the BS specifying their required M and L. We assume no collisions in this process, which is reasonable given the short length of these packets. Then, BS allocates transmissions slots for each node and broadcasts this information to the nodes. A node sends its packets in the designated slots and goes into sleep/idle mode for the other slots. Since BS coordinates the transmissions and each node transmits only in its allocated slots, there are no collisions. If a node does not receive the information about the allocated slots from the BS, it goes into sleep mode and retries the transmission at the next BS re-synchronization packet. Also, the BS can adapt the ideal power transmission of each active sensor node based on the signal strength attenuation, and sends this information in the broadcast message. A node sets up its transmission power to the one computed by the BS. This power is stored in sensor's memory for future transmissions. We define a round as the event

of transmission of packets by nodes in the network to the BS followed by an *Ack*. The total number of packets sent by S_i is on average $\frac{M}{1-e_i}$, which can be optimized for a minimum energy consumption as in [40]. The process ends successfully when the BS has received all the packets from all nodes. We assume that the BS has less energy consumption restrictions than the sensors, e.g., it is not battery powered. Thus, the total energy model shall exclude the energy consumption of the BS for now to focus on the more critical operation of the sensors. The following two protocols are considered:
(1) TNC: We assume that the BS knows e_i of each channel and thus can calculate the number of lost packets sent by node i (e.g., $R_i = M\frac{e_i}{1-e_i}$). We use a systematic network coding approach as it is described in [108]. The redundant packets are generated with RLNC, where each coded packet is an independent linear combination with high probability and a sufficiently large field size. The sensor node appends the coefficients used for the linear combination of each packet to that packet's header, so that the BS is able to decode. The total number of coded packets is denoted by $R = \sum_{i=1}^{N}\sum_{j=1}^{K} R_{i_j} = \sum_{i=1}^{N} R_i$, where R_{i_j} refers to the number of redundant packets sent by S_i in round j, K is the total number of rounds and R_i is the total number of redundant packets sent by S_i. To each node i, $M + R_i$ slots (M for uncoded packets and R_i for redundant packets) are allocated, where R_i is assumed to be sufficient to recover the lost data during the transmission. For simplicity, assume that the radio is turned off when the μC generates coded packets.
(2) TARQ: In the first round, each node sends its M packets in the allocated slots. After that, the BS acknowledges the received packets and allocates slots to each node for the transmission of the missing packets in the next round. This process is repeated until all data is recovered.

4.4 Numerical Results

We consider the limitation size for the data packet and *Ack*. Nodes are assumed to use the 802.15.4 protocol [109] for communication, which considers that the PHY packet has a payload of at most 127 bytes with 6 additional bytes used for the protocol's header, whereas the MAC packet size is 114 bytes plus 13 for the overhead (short addressing and unsecured frame). The *Ack* consists of 11 bytes. Simulation results (4.4.1) and real-life measurements (4.4.2) are provided for the models from Section 4.3.



(a) (b)

Figure 4.3: Coding time and transmit time (a) T_{ct} for $R = 1$ ($M = 5$ packets), different field sizes: (i) $T_{ict} = 125$ ns, $B = 8$, $C = 6$ and (ii) $T_{ict} = 62.5$ ns, $B = 16, C = 1$, (b) T_{ct} and T_t for $R = 1$ ($M = 5$ packets), different clock cycles, $\mathbf{GF}(2^{16})$, $T_{ict} = 125$ ns, $B = 8$ and rate= 250 kbps.

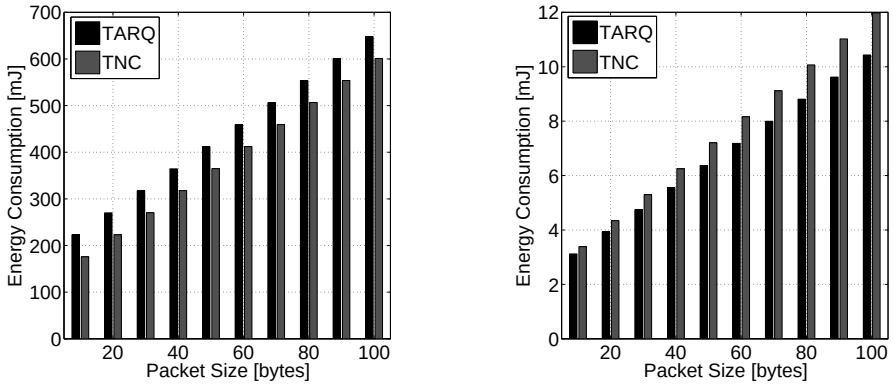
4.4.1 Dedicated/Reconfigurable Computing Model

4.4.1.1 Total Coding Time

We present this metric using two extreme scenarios for $R = 1$ and $M = 5$: (i) the case of a low performance μC with $T_{ict} = 125$ ns, $B = 8$, $C = 6$, and (ii) the case of a higher performance μC with $T_{ict} = 62.5$ ns, $B = 16$, $C = 1$ (see Figure 4.3(a)). For case (i) the time for computation T_{ct} for different field sizes and the packet sizes is significant, while for case (ii) all the T_{ct} values are below 1 ms. Note also that T_t for one packet is similar to T_{ct} for one coded packet when one instruction takes between 1 and 2 cycles to be executed (Figure 4.3(b)). Naturally, T_{ct} and T_t contribution to the E_{total} depends also on the parameters β and α, respectively. These two cases show that the choice of the appropriate hardware parameters (e.g., T_{ict}, B, C) is essential for a maximum energy efficiency of the communication protocols, specially those that require processing of the data.

4.4.1.2 Energy Consumption

The energy consumption is provided when all the nodes use the same platform and the protocols are applied for two classes of sensors, a Waspmote with 8-bit ATmega μC and a Zolertia with 16-bit Texas Instruments μC. We average the value for C using the related reference for Atmel [110] and Texas Instruments [111]. The switching time between active and sleep modes



(a) Waspmote XSC mote (b) Zolertia mote

Figure 4.4: Energy consumption of TNC and TARQ for $N = 2$ nodes, erasure probability $e_1 = 0.05$, $e_2 = 0.8$, $M = 5$ packets, $\mathbf{GF}(2^{16})$: (a) Waspmote XSC mote ($\alpha = 2.28$, $\beta = 0.21$, $\gamma = 1$, $\epsilon = 0.01$, $C = 1$, $B = 8$, $T_{ict} = 62.5$ ns, rate= 9.6 kbps) and (b) Zolertia mote ($\alpha = 1.08$, $\beta = 0.52$, $\gamma = 0.022$, $\epsilon = 0.001$, $C = 4$, $B = 16$, $T_{ict} = 62.5$ ns, rate= 250 kbps).

according with the specifications is small and the start-up time is negligible. We compute the energy consumption for both protocols and show the findings for $T_c = T_{ct}$, $M = 5$, $N = 2$, $\mathbf{GF}(2^{16})$ and asymmetric channels in Figure 4.4. The results illustrate that, for this case, TNC protocol performs better than TARQ for Waspmote platform and worst for the Zolertia model. Thus, the energy efficiency for a specific protocol may vary significantly for different sensor platforms.

4.4.2 Software Description Model

4.4.2.1 Network coding in Real-Life Measurements

We use TelosB motes (UC Berkeley, Crossbow) running TinyOS 2.1.1 operating system. This mote is described by a CC2420 radio using the IEEE 802.15.4 frame format and a MSP430F1611 μC. A Serial Peripheral Interface (SPI) connects the μC (master mode) and the radio (slave mode). A FIFO buffer is available at the radio for transmission, where one packet is loaded at the moment of transmission. The packet is first transferred from the μC through SPI and loaded into the FIFO buffer. The SPI resources are released after the complete load of the packet into the FIFO buffer. We define T_{SPI} as the time it takes for a packet from the moment the μC initiates the transmission and until the SPI resources are released. The transmit time is

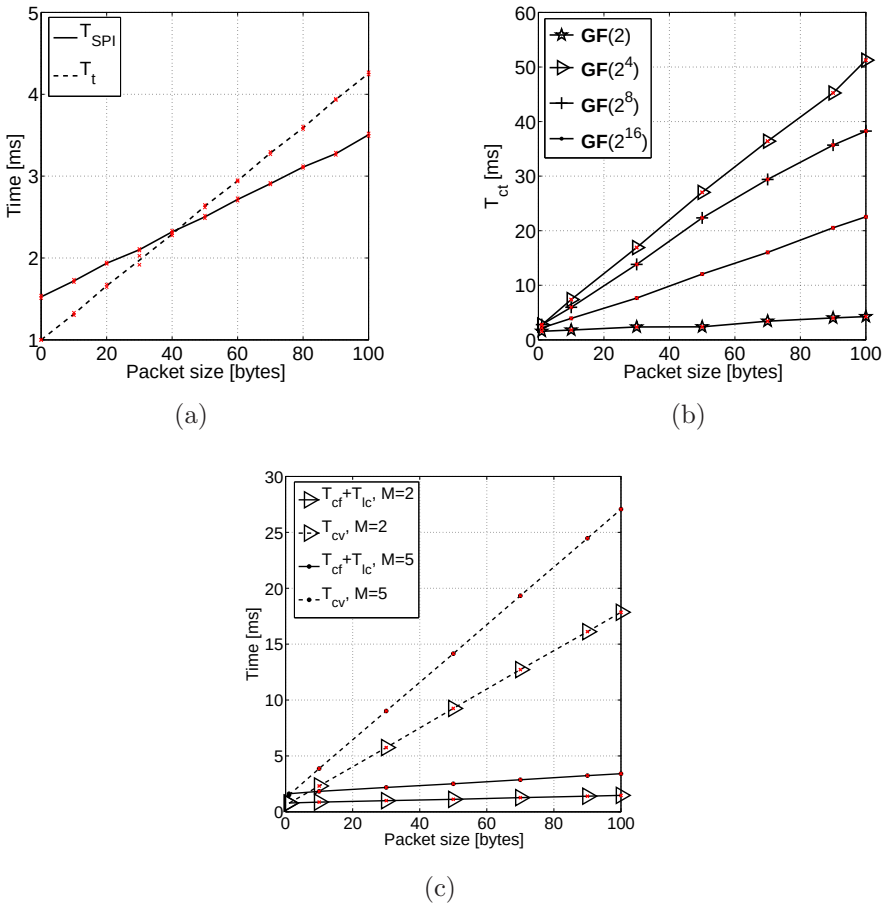


Figure 4.5: Transmit time and coding time using experimental results (a) T_{SPI} and T_t for one packet of different sizes, (b) Performing one linear combination among $M = 5$ packets for different field sizes, (c) Comparison of the time $T_{lc} + T_{cf}$ with T_{cv} for $R = 1$, $\mathbf{GF}(2^4)$, $M = 2$ and $M = 5$ packets.

the time after the SPI resources are released until the packet is sent through the air. The direct-memory access controller is enabled and the carrier sense multiple access protocol is disabled. Thus, the packets are transferred faster between μC and the radio and there is no need for the listening to the channel for collisions, because according with our protocols only motes in different transmission range send at a moment. We are interested on measuring the time to transmit an uncoded/coded packet (T_t), the time to code a packet (T_{ct}) and the time to receive an *Ack* (T_r).

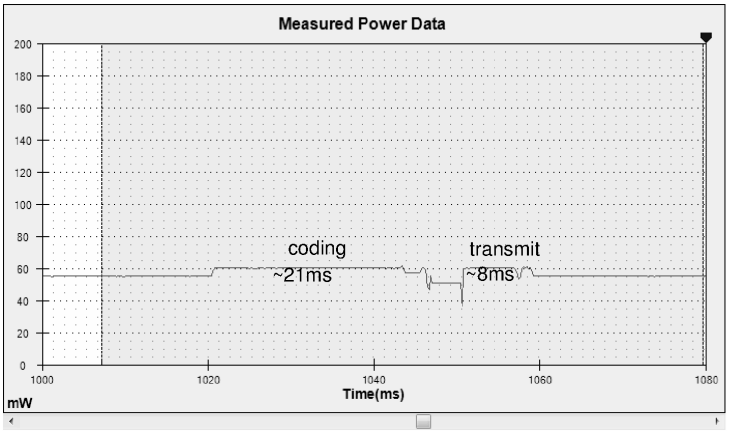


Figure 4.6: Power levels and active intervals for coding and transmission using Power Monitor device.

4.4.2.2 Measurements

We obtain the results by using the TinyOS timer module that can track the time a sensor spends while performing a specific operation. Therefore, a timer counter is used for all time measurements. We also use the Power Monitor device provided by the Monsoon Solutions Inc. [112] for a better accuracy of the measured values. We considered two TelosB motes communicating via 802.15.4 and connected to the computer through USB for recovering the measurements. Our results constitute an average of 3500 events and we provide the 97.5% confidence intervals for those measurements.

We perform the following experiments. (1) We first measure the transmission time of an uncoded packet. For this, we vary the size of the packet and the findings are plotted for T_{SPI} and T_t in Figure 4.5(a). For each addition of 10 bytes to the packet size, a linear increase of around 0.20 ms is observed for T_{SPI} and 0.30 ms for T_t. Also, the time for receiving an *Ack* is 2.1 ms. (2) We measure the coding time, i.e., the time to obtain one linear combination out of $M = 5$ packets, where the size of a packet is varied from 1 to 100 bytes. The results are shown in Figure 4.5(b), where T_{ct} is very high for $\mathbf{GF}(2^4)$ because of the conversion functions required. (3) Moreover, we observe that the time T_{cv} is much higher than T_{cf} together with T_{lc}. In this sense, we plot the results for $\mathbf{GF}(2^4)$, $M = 2$, $M = 5$ and different packet sizes in Figure 4.5(c). Thus, the conversion functions, which are not common RLNC operations, clearly limit the utilization of $\mathbf{GF}(2^4)$ in practice. In addition, the time to generate a coded packet out of $M = 5$ for $\mathbf{GF}(2^4)$ is reasonably small compared with $T_t + T_{SPI}$. (4) Also, we observe the additional time for sending one coded packet instead of an uncoded packet described in (1) can

increase significantly for large L, m, M, e.g., if $L = 50$ bytes and $m = 16$ are fixed and M is varied between 2 and 8, then a 22% increase is observed. Figure 4.6 illustrates the values for the power levels and the active periods obtained with the Power Monitor device for coding of a 100 bytes payload packet out of 5 packets, using $\mathbf{GF}(2^8)$, and transmission of the coded packet. These results validate the findings obtained using the timer module and are essential when designing protocols for a maximum energy efficiency in WSN, because they give insight on the need for judicious matching of the communication protocol to the sensor hardware, e.g., by using of field sizes that avoid the conversion functions.

4.5 Concluding Remarks

We analyzed relevant hardware characteristics of various sensors revealing that the transmission energy is not the main source of energy consumption for the majority of sensor platforms. Our analysis evidenced the need of a total energy consumption model to serve as a more accurate hardware abstraction for protocol design. Our preliminary results indicate that, by leveraging our proposed hardware abstractions, energy efficiency can be significantly improved either by (i) developing hardware-adaptive protocols, or (ii) choosing standard protocols judiciously in order to match the underlying hardware. Real-life measurements on TelosB motes support the latter. As part of the protocol design optimization, network coding has a full control for the number of redundant packets which can be updated online given the appropriate hardware parameters and the losses in the channel.

Chapter 5

Hardware-Aware Protocol Optimization

5.1 Motivation

As mentioned in Chapter 4, the design of the communication protocol for energy efficient applications needs to consider other power consumption, beside the transmission power, such as data processing, reception, switching, sleeping, and listening. However, in WSN we need to take into account that: (a) due to the low complexity of a sensor, a node can perform only one operation at a time; (b) transmission time can be greater or less than the one for processing; (c) sensor operations cannot be split, e.g., processing or transmission of a data packet; (d) the transmitted data can be lost; (e) sensors within the same radio range have to transmit the data at different moments in time to avoid interference; (f) the processing of the data by a node can be done simultaneously with the transmission by another node; and (g) the data is transmitted only after it is first encoded. The designer of the communication protocols in WSN calls for the integration of the sensor's capabilities, the effects of the transmission channel, and the requirements imposed by the application itself. Thus, the goal of this chapter is to offer ways to optimize a protocol based on the underlying hardware abstraction presented in Chapter 4.

The hardware abstraction model proposed in the previous chapter evaluates the total energy consumption, but optimum scheduling techniques of different operations at the sensors are essential. The challenge here is imposed by how we devise mechanisms to find a match between the protocol and the hardware for a low energy budget. The optimization aims on techniques that are developed for feedback based protocols and allowed the de-

signer to choose the communication protocol that spends less energy, given the challenges in the communication process. This is done by minimizing the energy consumption incurred due to different power consumption modes, e.g. transmission, processing, receiving, listening, and sleeping and also to different types of feedback, i.e., minimal feedback, meaning to signal the end of transmission of an entire batch of packets, intermediate feedback and feedback on a per-packet basis. Each communication protocol implemented here uses network coding techniques and the optimization is based on a Markov Decision Process. Thus, our main contributions are:

- *Optimizing the protocol to match the hardware* shows the analysis of the various types of feedback and channel conditions for different platforms.
- *Optimizing the hardware and the protocol jointly* demonstrates that a judicious match for the hardware of the sensors and the relay nodes is key to an energy efficient design.
- *Cross-validation of the results* based on real-life implementation of the protocols using TelosB motes.

The remainder of the chapter is organized as follows. An overview of the related work is provided in section 5.2 and the protocol optimization is provided in Section 5.3. The numerical results are given in Section 5.4 and the findings from the real-life measurements in Section 5.5. The chapter concludes with Section 5.6.

5.2 Related Work

As mentioned in Chapter 1 and Chapter 4, there exists significant literature focused on the minimization of communication budget in WSN, especially using the sensor's radio characteristics. Some more examples on top of the those already given are in [113], [114], [115], and [116]. Along with the radio characteristics, [113] also includes an energy analysis for the multi and single-hop transmission model, while [114] provides the energy consumption based on clustering mechanisms. [115] reeals solutions for the slot allocation mechanisms, with a low protocol overhead. Moreover, minimizing the lifetime of the sensor by providing a power-efficient MAC layer protocol is presented in [116]. Also, adaptive sampling techniques are provided in [117] in order to reduce the total amount of acquired data, and, thus, also to decrease the energy consumed for data communication. For biosignal monitoring sensors the main challenges to increase the energy efficiency are discussed in [118], considering the radio block (reception and transmission) and antenna. Typically,

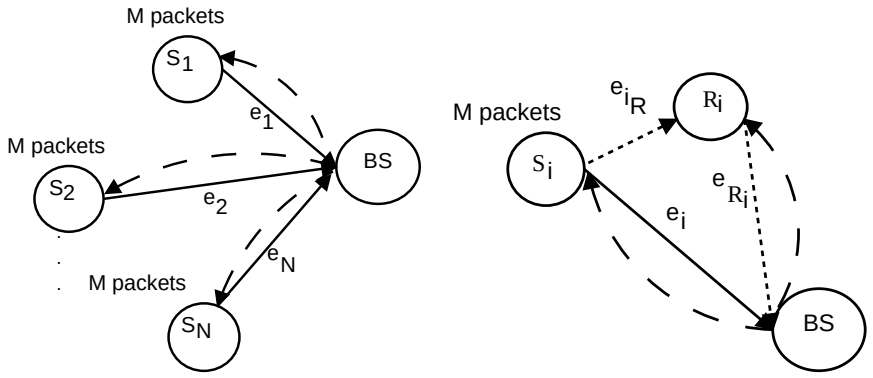
the communication protocols are not optimized in terms of energy based on both the hardware abstraction of the radio and the μC for any type of sensor platform. We provide such analysis, including also the feedback techniques and the cooperation with the relay nodes.

5.3 Problem Statement and Protocol Description

The previous results from Chapter 4 indicate that, by leveraging our proposed hardware abstraction, the energy efficiency can be significantly improved by either (i) developing hardware-adaptive protocols, or (ii) choosing standard protocols judiciously in order to match the underlying hardware. As part of the protocol design optimization, network coding has a full control of the number of redundant packets which can be updated online given the appropriate hardware parameters and the losses in the channel. Hence, assuming that the transmission, processing, receiving, listening, and sleeping time are known as well as the channel losses and, given the limitations of the communication process and the sensor's hardware, our goal is to optimize the communication protocol. This can be done by the mean of making the best decision in each moment of time among the following options (a) coding the data before sending it, (b) transmission of the coded packets, (c) listening to the channel for reception of an acknowledgement packet, or (d) sleeping, in such a way that the energy consumption is minimized, while gathering the data at the base station (BS). Thus, the performance metric is the total energy consumption of the system, which is computed using the expression in (4.1). In the following, we describe the assumptions used, our basic setup, the protocols and the main result.

5.3.1 Assumptions and Network Setup

We start by enumerating the set of conflicts considered significant for this part of the work. First, the sensors are very simple devices, and, hence, we assume that (1) a node can perform only one operation at a time (e.g., it cannot process and transmit in the same time), (2) the time for transmission can be greater or less than the one for processing, and (3) sensor operations cannot be split, e.g., processing or transmitting of a data packet, except that external factors are involved, like the depletion of the battery. Second, the communication process introduces some challenges for the channel and sensors, and thus, (4) the transmitted data can be lost and (5) the sensors within the same radio range have to transmit the data at different moments



(a) N nodes (b) 1 node

Figure 5.1: System model: (a) Star topology; (b) Star-plus topology.

in time to avoid interference. Third, some rules might be imposed by the designer in order to improve the efficiency of the communication protocol, such as (6) the processing of the data by a node can be done simultaneously with the transmission by another node, and, depending on the type of data, (7) a packet is transmitted only after it is first encoded.

We consider a N-node *star topology* network, where a sensor node, denoted by S_i, $\forall i = \{1, 2, \ldots, N\}$, wants to send directly M packets, each one of L bytes, to a BS (Figure 5.1(a)). In addition, a *star-plus topology* is also assumed, where each node can send data directly to a BS or through relay nodes R_i (Figure 5.1(b)). The channels between the sensors and BS are characterized by the erasure probability e_i, between the sensors and relay by e_{i_R}, and between the relay and the BS by e_{R_i}. Each packet is sent in a time slot and BS acknowledges the nodes about the reception of the packets. Each acknowledgement packet has Ack bytes and is successfully received by all nodes. The direct transmission from the nodes to the BS is represented by solid lines, the feedback from the BS to the nodes by dashed lines, while the transmissions related to the relay by small-dashed lines, as Figure 5.1 shows. Also, we assume that only the BS sends feedback about the reception of the packets, while the relay is not able to send any type of acknowledgement. Both the relay and the source node are able to perform the same operations, like processing, transmitting, listening, receiving, and sleeping.

At the beginning, BS broadcasts a beacon message to synchronize the nodes. Active nodes send a short length message to BS specifying their required M and L. Then, BS allocates transmissions slots for each node and broadcasts this information to the nodes. A node sends its packets in the designated slots and goes into sleep/idle mode for the other slots. If a node does not receive

the information about the allocated slots from BS, it goes into sleep mode and retries the transmission at the next BS re-synchronization packet. We define a round as the event of transmission of packets by nodes in the network to BS followed by an acknowledgement. The total number of packets sent by S_i is on average $\frac{M}{1-e_i}$, which can be optimized for a minimum energy consumption as in [40]. The process ends successfully when BS has received all the packets from all nodes. We assume that BS has less energy consumption restrictions than the sensors, e.g., it is not battery powered. Thus, the total energy model shall exclude the energy consumption of BS for now to focus on the more critical operation of the sensors.
We describe 3 protocols for gathering the data from the nodes, which are different from the point of view of the feedback sent by BS to the sensors to acknowledge the reception of the packets. (a) $No\ feedback$ means that exists no feedback between the transmissions, only one acknowledgement signalling the end of transmission process. Here, a packet is discarded from the buffer after each transmission, whether it was successfully transmitted or not. (b) $Full\ feedback$ applies when the BS announces the node about the reception of the packet after each transmission, such that an unsuccessfully transmitted packet is kept in the buffer of the node until it is correctly received at the BS. These two protocols consider simple actions, such as sleeping, coding, transmitting or receiving. (c) $Min\ feedback$ refers to a protocol where an action is defined by some sets of joint simple actions. A feedback is valid if in the joint actions at least one transmission occurs, and a packet is discarded whether it was successfully or not transmitted. For all cases, we assume that the feedback cannot be lost or delayed, but introduces costs for listening and receiving.

5.3.2 Main Result

The data is represented by packets of the same length and the coding and transmission process are modelled by using queues, as Figure 5.2 shows. For the source node, one queue is used to stored the original packets and one for the coded packets obtained using RLNC. The queue for the coded packets is denoted by q_i and its length is $Y \geq 1$. For the relay, one queue is necessary to keep the received coded packets from node S_i, denoted by $q^1_{R_i}$ and one for recoding the packets from queue $q^1_{R_i}$, called $q^2_{R_i}$. We assume the size of $q^1_{R_i}$ and $q^2_{R_i}$ are equal to the one of q_i. Another queue q_{BS_i} is defined at the BS for the received coded packets from each S_i. The size is $q_{BS_i} \leq M$, because the coded packets are linearly independent with high probability, since the Galois field used to perform the coded packets is high enough [53].
The model can be described by using a MDP $\{(Q(n), A(n)), n \geq 0\}$, where

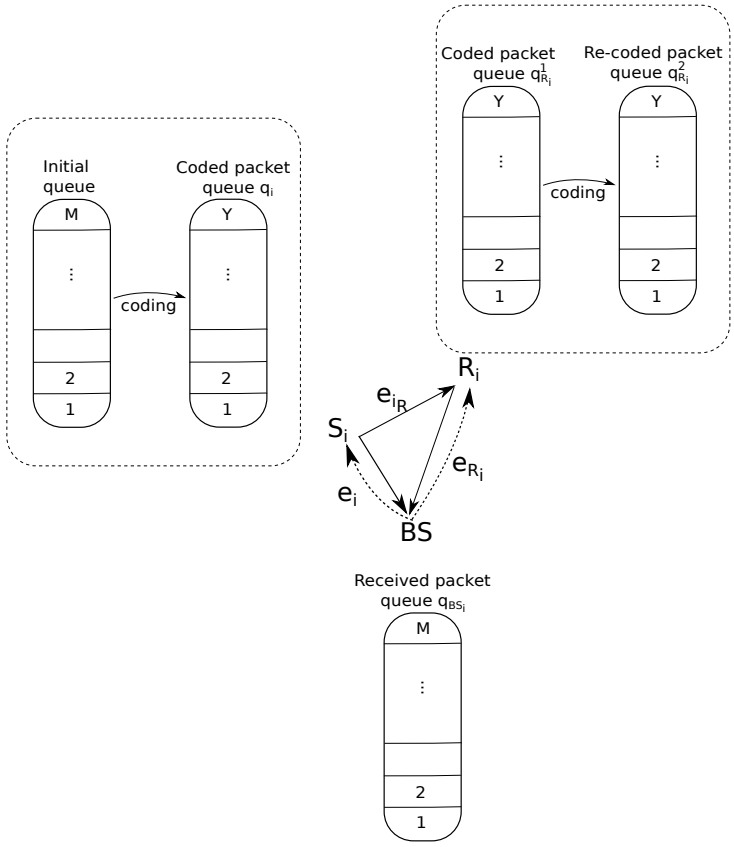
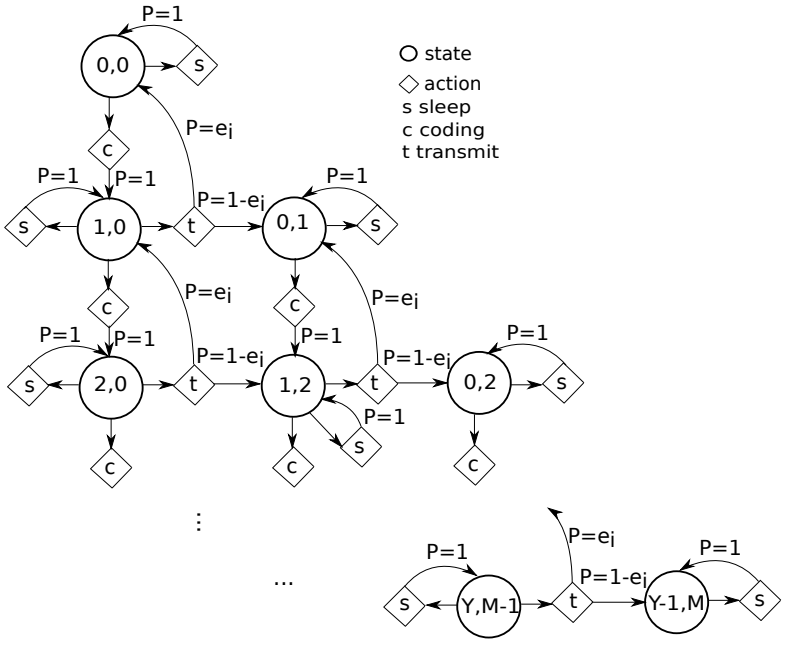
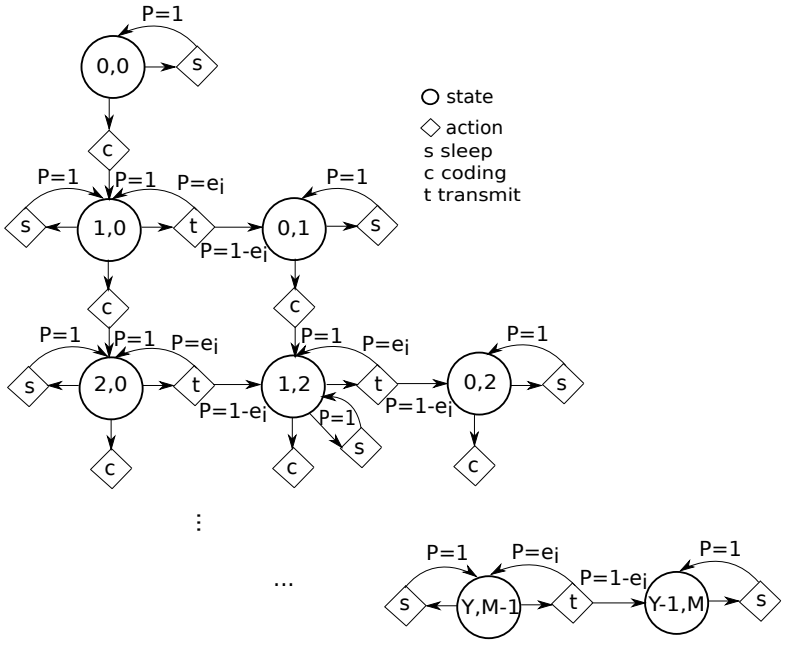


Figure 5.2: Queueing model for one node, one relay, and the BS.

n represents the time slot, $Q(n)$ is the state of the system, and $A(n)$ is the action taken at time n for state $Q(n)$. The state $Q(n)$ means the joint state, while for each source node the state description is given by the queue q_i, for a relay node by $q^1_{R_i}$, $q^2_{R_i}$, and for the BS by q_{BS_i}. The notation used for action $A(n)$ refers also to the joint action of the nodes, while the action for each source node is given by $a_i(n)$, and for each relay by a_{R_i}, $\forall i = \{1, 2, ..., N\}$. Furthermore, we denote the cost for action $A(n)$ being in state $Q(n)$ at time slot n by $C(Q(n), A(n))$. The cost related to an action for S_i is expressed using the active periods T^i_s, T^i_c, T^i_t, T^i_r, and T^i_l, where T^i_s means the time for sleeping, T^i_c for processing, T^i_t for transmission, T^i_r for receiving a packet or an acknowledgement, and T^i_l for listening for a packet or a feedback. The costs for listening and receiving of a feedback are added to the final cost. Sleeping actions should be used in a restricted manner because they delay the completion of the process and waste energy. To model this, penalties are enforced for the sleeping action. Only the last states of the model are exempted from this rule. Further, the transition from the current state $Q(n)$



(a)



(b)

Figure 5.3: MDP for one node in a star topology and finite number of transmitted packets: (a) *no feedback*, (b) *full feedback*.

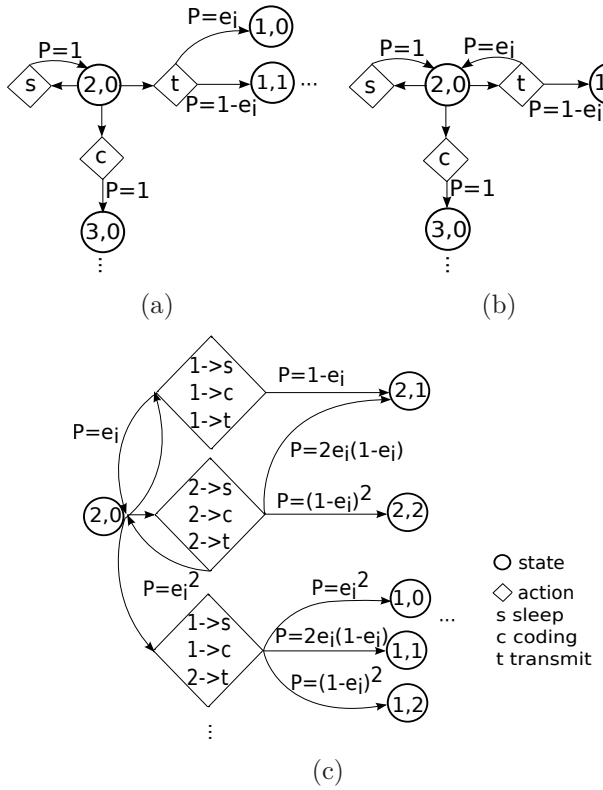


Figure 5.4: MDP example for one node in star topology when the current state is $(2,0)$, $M > 2$ and $Y > 2$: (a) *no feedback*, (b) *full feedback* and (c) *min feedback*.

at time n to the next state $Q(n+1)$ given the action $A(n)$ is defined by the transition probability $\mathcal{P}_{Q(n)Q(n+1)}(A(n))$. The complete energy budget is easily derived by using the fixed parameters $\alpha, \beta, \gamma, \epsilon, P_t$ from (4.1). For a better understanding, we provide a description in Figure 5.3 with all possible transitions for *no feedback* and *full feedback* scheme from the first to the last state of the model. Note that the same thing is difficult to show for the *min feedback* scheme, since there are infinite possible cases and the choice of the joint actions is dependent on each application developer. The detailed description for the states, actions, costs, and transition probability for each protocol is given after the example below. We start by describing the case of one simple node, for a star and a star-plus topology, and at the end we generalize the model to multiple nodes.

Figure 5.4 shows an example for *no feedback*, *full feedback*, and *min feedback*, respectively, for the case of one node whose current state is $(2,0)$, $M > 2$, and $Y > 2$. The representation includes states, shown by solid line circles, actions

by diamond shapes, and transitions between states. In case of *no feedback*, state $(2,0)$ transits to $(1,0)$ if the transmission is unsuccessful, and to $(1,1)$ for successful transmission. For coding action, the transition is possible with probability 1 if both queues at the node and the BS are not full yet. Sleeping is possible in any state with probability 1. A similar situation occurs for *full feedback*, the only exception is related to the unsuccessful transmission, where the packet is not discarded from the buffer. For both, *no feedback* and *full feedback*, the figure shows all the possible transitions from state $(2,0)$. But, in case of *min feedback* we show only 3 sets of actions, i.e., $(x_i = 1, y_i = 1, z_i = 1)$, $(x_i = 2, y_i = 2, z_i = 2)$, and $(x_i = 1, y_i = 1, z_i = 2)$, but other sets are possible. Here, the transition to future states depends on both coding and transmission operations from the joint actions.

5.3.2.1 Star Topology for N = 1

The state of the model is given by $Q(n) = (q_i(n), q_{BS_i}(n))$, where q_i gives information about the available space for new coded packets in the buffer of S_i, and also the available number of coded packets for the transmission. Moreover, q_{BS_i} provides information about the required packets at the BS. The first state of the model is $(0,0)$ and the process stops when at least one of the absorption states of the model is reached, i.e., $q_{BS_i} = M$. The state space Θ is given by all possible combinations of the form $Q(n)$, with $\Theta = \{(q_i(n), q_{BS_i}(n)) : 0 \leq q_i(n) \leq Y \text{ and } 0 \leq q_{BS_i}(n) \leq M\} \setminus \{(q_i(n) = Y, q_{BS_i}(n) = M)\}$. State $(q_i(n) = Y, q_{BS_i}(n) = M)$ is not valid since one node cannot process, transmit, and/or receive in the same time. The actions, costs and transition probabilities for *no feedback*, *full feedback*, and *min feedback* are described in the following.

No feedback and *full feedback* Protocol

The action set for a node S_i, $\forall i = \{1, 2, ..., N\}$ is represented by $A(n) = \{a_i(n) \in \{s, c, t\}, \text{with } s = sleeping, c = coding, t = transmit\}$, where a node can take at each moment one action from this set. A coding action is valid if $q_i(n) < Y$ and $q_{BS_i}(n) < M$, meaning that the node's queue is not full and less than M packets have been received at the BS from node S_i. Also, a transmission action is possible if $q_i(n) > 0$ and $q_{BS_i}(n) < M$, where the first condition means that the coded queue at node S_i should contain at least one coded packet. The cost for action $A(n)$ being in state $Q(n)$ at time

slot n is described by

$$C(Q(n), A(n)) = \begin{cases} T_s^i, & \text{if } a_i(n) = s; \\ T_c^i, & \text{if } a_i(n) = c; \\ T_t^i, & \text{if } a_i(n) = t; \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

To transit from the current state $Q(n) = (q_i(n), q_{BS_i}(n))$ at time n to the next state $Q(n+1) = (q_i(n+1), q_{BS_i}(n+1))$ given the action $A(n)$, we define the transition probability below, including also the changes in the queues for each case,

$$\mathcal{P}_{Q(n)Q(n+1)}(A(n)) = \begin{cases} 1, \text{if } a_i(n) = s \text{ (sleeping)}, \\ q_{BS_i}(n+1) = q_{BS_i}(n), q_i(n+1) = q_i(n), \\ \text{or if } a_i(n) = c \text{ (coding)}, \\ q_{BS_i}(n+1) = q_{BS_i}(n), q_i(n+1) = q_i(n) + 1; \\ 1 - e_i, \text{if } a_i(n) = t \text{ (successful transmission)}, \\ q_{BS_i}(n+1) = q_{BS_i}(n) + 1, q_i(n+1) = q_i(n) - 1; \\ e_i, \text{if } a_i(n) = t \text{ (unsuccessful transmission)}, \\ q_{BS_i}(n+1) = q_{BS_i}(n), \\ q_i(n+1) = q_i(n) - 1 \ (No\ feedback), \\ q_i(n+1) = q_i(n) \ (Full\ feedback); \\ 0, \text{ otherwise.} \end{cases} \tag{5.2}$$

Min feedback Protocol

The action set is given by $A(n) = \{a_i(n) = (x_i \to s, y_i \to c, z_i \to t), \text{with } x_i, y_i, z_i \in \mathbb{N}, y_i \leq Y, z_i \leq M\}$, which means S_i can take the following actions sequentially, not necessarily in this order: sleep x_i time slots, perform y_i coded packets, and transmit z_i packets. The cost is given by $C(Q(n), A(n)) = x_i T_s^i + y_i T_c^i + z_i T_t^i$. A multiple action that includes coding and transmission is valid when $q_i(n) + y_i \leq Y$, $z_i \leq q_i(n)$, and $q_{BS_i}(n) < M$. The transition probabilities are

$$\mathcal{P}_{Q(n)Q(n+1)}(A(n)) = \begin{cases} \mathcal{P}_S, \text{if } a_i(n) = (x_i \to s, y_i \to c, z_i \to t) \\ \text{(sleeping, coding, and/or transmitting),} \\ q_{BS_i}(n+1) = q_{BS_i}(n) + \mathfrak{N}_{z_i}, \\ q_i(n+1) = q_i(n) + y_i - z_i; \\ 0, \text{ otherwise.} \end{cases} \tag{5.3}$$

$$\mathcal{P}_S = \binom{z_i}{q_{BS_i}(n+1) - q_{BS_i}(n)} \cdot \\ \cdot (e_i)^{(z_i - q_{BS_i}(n+1) + q_{BS_i}(n))} (1-e_i)^{(q_{BS_i}(n+1) - q_{BS_i}(n))}. \tag{5.4}$$

where $\mathfrak{N}_{z_i}$ packets from z_i are correctly received at the BS, with $\mathfrak{N}_{z_i} \in [0, 1, ..., z_i]$.

5.3.2.2 Star-Plus Topology for N = 1

The relay recodes the received packets from the node before the transmission, thus, the state of the MDP model is described by $Q(n) = (q_i(n), q^1_{R_i}(n), q^2_{R_i}(n), q_{BS_i}(n))$, with q_i, $q^1_{R_i}$, $q^2_{R_i}$, and q_{BS_i} as previously described. The state space Θ is given by all combinations of the form $Q(n)$, except $(q_i(n) = Y, q^1_{R_i}(n) = Y, q^2_{R_i}(n) = Y, q_{BS_i}(n) = M)$:

$$\Theta = \{(q_i(n), q^1_{R_i}(n), q^2_{R_i}(n), q_{BS_i}(n)) : 0 \leq q_i(n) \leq Y, 0 \leq q^1_{R_i}(n) \leq Y, 0 \leq q^2_{R_i}(n) \leq Y \\ \text{and } 0 \leq q_{BS_i}(n) \leq M\} \setminus \{(q_i(n) = Y, q^1_{R_i}(n) = Y, q^2_{R_i}(n) = Y, q_{BS_i}(n) = M)\}.$$

A packet is discarded from $q^1_{R_i}$ only after it is recoded and from $q^2_{R_i}$ after it is transmitted, according to the protocol used.

Table 5.1: Possible joint operations for node S_i and relay R_i in time slot n from the set $\{s = sleeping, c = coding, t = transmit, l/r = listen/receive\}$, with $i \in 1, 2, ...N$.

$a_i(n)$	t	t	c	c	c	s	s	s	l/r
$a_{R_i}(n)$	s	l/r	s	t	c	s	c	t	l/r

No feedback and *full feedback* Protocol

The action set is represented by $A(n) = \{(a_i(n), a_{R_i}(n)) : a_i(n) \in \{s, c, t, l/r\}, a_{R_i}(n) \in \{s, c, t, l/r\}\}$. According to the assumptions from 5.3.1 (1)-(7), we define a list with the possible operations at the node and relay in the same time slot (Table 5.1). The action $a_{R_i}(n) = l/r$ is available for the relay only when the node is transmitting and for both, the node and relay, when the BS sends acknowledgement packets. The cost function is

$$C(Q(n),A(n))=\begin{cases} T_t^i+T_s^{R_i}, & \text{if } a_i(n)=t, a_{R_i}(n)=s; \\ T_t^i+T_r^{R_i}+T_l^{R_i}, & \text{if } a_i(n)=t, a_{R_i}(n)=l/r; \\ T_c^i+T_s^{R_i}, & \text{if } a_i(n)=c, a_{R_i}(n)=s; \\ T_c^i+T_t^{R_i}, & \text{if } a_i(n)=c, a_{R_i}(n)=t; \\ T_c^i+T_c^{R_i}, & \text{if } a_i(n)=c, a_{R_i}(n)=c; \\ T_s^i+T_s^{R_i}, & \text{if } a_i(n)=s, a_{R_i}(n)=s; \\ T_s^i+T_c^{R_i}, & \text{if } a_i(n)=s, a_{R_i}(n)=c; \\ T_s^i+T_t^{R_i}, & \text{if } a_i(n)=s, a_{R_i}(n)=t; \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

A transmission action is valid for S_i when $q_i(n) > 0$ and $q_{BS_i}(n) < M$, where a coding action is possible if $q_i(n) < Y$ and $q_{BS_i}(n) < M$. The transmission is still valid if the relay's queue for received coded packets queue is full ($q^1_{R_i}(n) = Y$). The condition for a relay to transmit is imposed by $q^2_{R_i}(n) > 0$ meaning that the relay has available coded packets for transmission, and $q_{BS_i}(n) < M$. The coding operation at relay is possible if $q^1_{R_i}(n) > 1, q^2_{R_i}(n) < Y$, and $q_{BS_i}(n) < M$. The transition probability when only the node transmits is given by (5.6), and the transition probability when only the relay transmits is in (5.7),

$$\mathcal{P}_{Q(n)Q(n+1)}(A(n)) = \tag{5.6}$$

$$\begin{cases}
1, & \text{if } a_i(n)=s, a_{R_i}(n)=s \text{ (sleeping), } q_{BS_i}(n+1)=q_{BS_i}(n), \\
& q_i(n+1)=q_i(n), q^1_{R_i}(n+1)=q^1_{R_i}(n), q^2_{R_i}(n+1)=q^2_{R_i}(n) \\
& \text{or if } a_i(n)=c, a_{R_i}(n)=c \text{ (coding at } S_i \text{ and } R_i), \\
& q_{BS_i}(n+1)=q_{BS_i}(n), q_i(n+1)=q_i(n)+1, \\
& q^1_{R_i}(n+1)=q^1_{R_i}(n)-1, q^2_{R_i}(n+1)=q^2_{R_i}(n)+1 \\
& \text{or if } a_i(n)=c, a_{R_i}(n)=s \text{ (coding at } S_i), \\
& q_{BS_i}(n+1)=q_{BS_i}(n), q_i(n+1)=q_i(n)+1, \\
& q^1_{R_i}(n+1)=q^1_{R_i}(n), q^2_{R_i}(n+1)=q^2_{R_i}(n) \\
& \text{or if } a_i(n)=s, a_{R_i}(n)=c \text{ (coding at } R_i), \\
& q_{BS_i}(n+1)=q_{BS_i}(n), q_i(n+1)=q_i(n), \\
& q^1_{R_i}(n+1)=q^1_{R_i}(n)-1, q^2_{R_i}(n+1)=q^2_{R_i}(n)+1; \\
(1-e_i)(1-e_{i_R}), & \text{if } a_i(n)=t, a_{R_i}(n)=l/r \text{ (successful transmission for } BS \text{ and } R_i), \\
& q_{BS_i}(n+1)=q_{BS_i}(n)+1, q_i(n+1)=q_i(n)-1, \\
& q^1_{R_i}(n+1)=q^1_{R_i}(n)+1, q^2_{R_i}(n+1)=q^2_{R_i}(n); \\
(1-e_i)e_{i_R}, & \text{if } a_i(n)=t, a_{R_i}(n)=s \text{ or } a_i(n)=t, a_{R_i}(n)=l/r \text{ (successful} \\
& \text{transmission for } BS), q_{BS_i}(n+1)=q_{BS_i}(n)+1, \\
& q_i(n+1)=q_i(n)-1, q^1_{R_i}(n+1)=q^1_{R_i}(n), q^2_{R_i}(n+1)=q^2_{R_i}(n); \\
e_i(1-e_{i_R}), & \text{if } a_i(n)=t, a_{R_i}(n)=l/r \text{ (successful transmission for } R_i), \\
& q_{BS_i}(n+1)=q_{BS_i}(n), q^1_{R_i}(n+1)=q^1_{R_i}(n)+1, q^2_{R_i}(n+1)=q^2_{R_i}(n), \\
& q_i(n+1)=q_i(n)-1 (No\ feedback), \\
& q_i(n+1)=q_i(n) (Full\ feedback); \\
e_i e_{i_R}, & \text{if } a_i(n)=t, a_{R_i}(n)=s \text{ or } a_i(n)=t, a_{R_i}(n)=l/r \text{ (unsuccessful} \\
& \text{transmission for } BS \text{ and } R_i), \\
& q_{BS_i}(n+1)=q_{BS_i}(n), q^1_{R_i}(n+1)=q^1_{R_i}(n), \\
& q^2_{R_i}(n+1)=q^2_{R_i}(n), \\
& q_i(n+1)=q_i(n)-1 \ (No\ feedback), \\
& q_i(n+1)=q_i(n) \ (Full\ feedback); \\
0, & \text{otherwise.}
\end{cases}$$

$$\mathcal{P}_{Q(n)Q(n+1)}(A(n)) = \tag{5.7}$$
$$\begin{cases} 1-e_{R_i}, \text{ if } a_i(n)=c, a_{R_i}(n)=t \text{ or } a_i(n)=s, a_{R_i}(n)=t \\ \qquad \text{(successful transmission for } BS), \\ \qquad q_{BS_i}(n+1)=q_{BS_i}(n)+1, \\ \qquad q_i(n+1)=q_i(n), q^1_{R_i}(n+1)=q^1_{R_i}(n), \\ \qquad q^2_{R_i}(n+1)=q^2_{R_i}(n)-1; \\ e_{R_i}, \quad \text{if } a_i(n)=c, a_{R_i}(n)=t \text{ or } a_i(n)=s, a_{R_i}(n)=t \\ \qquad \text{(unsuccessful transmission for } BS), \\ \qquad q_{BS_i}(n+1)=q_{BS_i}(n), \\ \qquad q_i(n+1)=q_i(n), q^1_{R_i}(n+1)=q^1_{R_i}(n), \\ \qquad q^2_{R_i}(n+1)=q^2_{R_i}(n)-1 \ (No\ feedback), \\ \qquad q^2_{R_i}(n+1)=q^2_{R_i}(n) \ (Full\ feedback); \\ 0, \qquad \text{otherwise.} \end{cases}$$

Min feedback Protocol

The combined actions are defined for the node and relay by

$$\begin{aligned} A(n) = \{(a_i(n), a_{R_i}(n)) = ((x_i \to s, y_i \to c, z_i \to t), \\ (x_{R_i} \to s, y_{R_i} \to c, z_{R_i} \to t, w_{R_i} \to l/r)), \\ \text{with } x_i, y_i, z_i, x_{Ri}, y_{Ri}, z_{R_i}, w_{R_i} \in \mathbb{N}, \\ y_i \le Y, y_{R_i} \le Y, z_i \le M, z_{R_i} \le M\}, \end{aligned} \tag{5.8}$$

but only some combinations are valid for each time slot, as Table 5.1 shows. The cost function is given by $C(Q(n), A(n)) = x_i T^i_s + y_i T^i_c + z_i T^i_t + x_{R_i} T^{R_i}_s + y_{R_i} T^{R_i}_c + z_{R_i} T^{R_i}_t + w_{R_i}(T^{R_i}_l + T^{R_i}_r)$. Here, $T^{R_i}_l$ and $T^{R_i}_r$ are related to listening and receiving for a packet transmitted from the source node. A multiple action that implies coding and transmission at source is possible when $q_i(n) + y_i \le Y, z_i \le q_i(n)$, and $q_{BS_i}(n) < M$. A transmission action at relay is valid when $z_{R_i} \le q^2_{R_i}(n), q_{BS_i}(n) < M$, and a coding operation when $q^1_{R_i}(n) > 1, q^2_{R_i}(n) + y_{R_i} \le Y$, and $q_{BS_i}(n) < M$.

Further, the transition probability when the source node transmits is

$$\mathcal{P}_{Q(n)Q(n+1)}(A(n)) = \tag{5.9}$$

$$\begin{cases} \mathcal{P}_{SR}\mathcal{P}_S, \text{if}\,(a_i(n), a_{R_i}(n)) = ((x_i \to s, y_i \to c, z_i \to t), \\ \quad (x_{R_i} \to s, y_{R_i} \to c, z_{R_i} \to 0, w_{R_i} \to l/r)), \\ \quad \text{(sleeping, coding, and/or transmitting)}, \\ \quad q_{BS_i}(n+1) = q_{BS_i}(n) + \mathfrak{N}_{z_i}, \\ \quad q_i(n+1) = q_i(n) + y_i - z_i, \\ \quad q^1_{R_i}(n+1) = q^1_{R_i}(n) + \mathfrak{N}^{R_i}_{z_i} - y_{R_i}, \\ \quad q^2_{R_i}(n+1) = q^2_{R_i}(n) + y_{R_i}; \\ 0, \text{otherwise} \end{cases} \tag{5.10}$$

$$\mathcal{P}_{SR} = \binom{z_i}{q^1_{R_i}(n+1) - q^1_{R_i}(n)} \cdot \tag{5.11}$$
$$\cdot (e_{i_R})^{(z_i - q^1_{R^i}(n+1) + q^1_{R_i}(n))} (1 - e_{i_R})^{(q^1_{R_i}(n+1) - q^1_{R_i}(n))},$$

with $\mathcal{P}_S$ in (5.4). The number of received packets from node S_i at the BS is given by $\mathfrak{N}_{z_i}$ and at the relay by $\mathfrak{N}^{R_i}_{z_i}$, with $\mathfrak{N}_{z_i}$ and $\mathfrak{N}^{R_i}_{z_i} \in [0, 1, ..., z_i]$.

In the case of a relay, the transition probability for a multiple action that implies transmission is given by

$$\mathcal{P}_{Q(n)Q(n+1)}(A(n)) = \tag{5.12}$$

$$\begin{cases} \mathcal{P}_R, \text{if}\,(a_i(n), a_{R_i}(n)) = ((x_i \to s, y_i \to c, z_i \to 0), \\ \quad (x_{R_i} \to s, y_{R_i} \to c, z_{R_i} \to t, w_{R_i} \to l/r)), \\ \quad \text{(sleeping, coding, and/or transmitting)}, \\ \quad q_{BS_i}(n+1) = q_{BS_i}(n) + \mathfrak{N}_{z_{R_i}}, \\ \quad q_i(n+1) = q_i(n) + y_i, q^1_{R_i}(n+1) = q^1_{R_i}(n), \\ \quad q^2_{R_i}(n+1) = q^2_{R_i}(n) - z_{R_i}; \\ 0, \text{ otherwise} \end{cases} \tag{5.13}$$

where

$$\mathcal{P}_R = \binom{z_{R_i}}{q_{BS_i}(n+1) - q_{BS_i}(n)} (e_{R_i})^{(z_{R_i} - q_{BS_i}(n+1) + q_{BS_i}(n))} \cdot$$
$$\cdot (1 - e_{R_i})^{(q_{BS_i}(n+1) - q_{BS_i}(n))}. \tag{5.14}$$

Table 5.2: Possible joint operations for node S_i and S_j in time slot n from the set $\{s = sleeping, c = coding, t = transmit, l/r = listen/receive\}$, with $i, j \in 1, 2, ...N$ and $i \neq j$.

$a_i(n)$	t	t	c	c	c	s	s	s	l/r
$a_j(n)$	s	c	s	t	c	s	c	t	l/r

From z_{R_i} packets transmitted, $\mathfrak{N}_{z_{R_i}} \in [0, 1, ..., z_{R_i}]$ are correctly received at the BS.

5.3.2.3 Star Topology for N>1

The state is described by $Q(n) = ((q_1(n), q_{BS_1}(n)), (q_2(n), q_{BS_2}(n))..., (q_N(n), q_{BS_N}(n)))$, while the state space by $\Theta = \{(q_1, q_{BS_1}), (q_2, q_{BS_2}), ..., (q_N, q_{BS_N})) : 0 \leq q_i \leq Y$ and $0 \leq q_{BS_i} \leq M\} \setminus \{(q_1 = Y, q_{BS_1} = M), ..., (q_N = Y, q_{BS_N} = M)\}$. The possible joint operations for two nodes are shown in Table 5.2.

***No feedback* and *full feedback* Protocol**

The action set is $A(n) = \{(a_1(n), a_2(n), ..., a_N(n)) : a_i(n) \in \{s, c, t, l/r\}$, where for $N > 2$ similar conditions as in Table 5.2 are applied. The costs associated to these actions are below. For simplicity, we denote in the equation (5.15) the cost $C(Q(n), a_i(n))$ for a node S_i by C_i. In addition to the rules defined for $N = 1$, we have the following situations:

$$C(Q(n), A(n)) = \begin{cases} \max_i(C_i), \text{if } a_i(n) = c, i \in 1, .., N; \\ \max(\max_i(C_i), C_j), \text{if } a_i(n) = c, a_j(n) = t, \\ i \in 1, .., N \setminus \{j\}; \\ C_j, \text{if } a_i(n) = s, a_j(n) = t, \\ i \in 1, .., N \setminus \{j\}; \\ \max_i(C_i), \text{if } a_i(n) = c, a_j(n) = s, \\ i \in 1, .., N \setminus \{j\}; \\ \min_i(C_i), \text{if } a_i(n) = s, i \in 1, .., N. \end{cases} \tag{5.15}$$

The transition probability from state $Q(n)$ to state $Q(n+1)$ for a given action $A(n)$ is

$$\mathcal{P}_{Q(n)Q(n+1)}(A(n)) = \prod_{i=1}^{N} \mathcal{P}_{(q_i(n),q_{BS_i}(n))(q_i(n+1),q_{BS_i}(n+1))}(a_i(n)), \tag{5.16}$$

where the probability $\mathcal{P}_{(q_i(n),q_{BS_i}(n))(q_i(n+1),q_{BS_i}(n+1))}\ (a_i(n))$ is characterized as in (5.2) for each node S_i.

*M*in feedback protocol

The action set is given by

$$\begin{aligned} A(n) &= \{(a_1(n), a_2(n), ..., a_N(n)) : a_i(n) = \\ &= \{(x_i \rightarrow s, y_i \rightarrow c, z_i \rightarrow t),\ \text{with } x_i, y_i, z_i \in \mathbb{N}\}\}. \end{aligned} \tag{5.17}$$

The operations for each slot should guarantee the assumptions from Table 5.2. In particular, the costs from (5.15) are examples when the joint combination is composed by just a simple action. In general, the cost function is given by Algorithm 4, considering the descending order of the coding time for each node, $y_i T_c^i$. The transition probabilities are given by (5.16), where actions are chosen as in (5.17).

Algorithm 4 Cost value C(Q(n),A(n)) of the MDP model for more than one node $N > 1$, *min feedback* protocol and $x_i, y_i, z_i \neq 0$

Data: T_s^i, T_c^i, T_t^i, protocol parameters x_i, y_i, z_i, $\forall i = \{1, 2, .., N\}$ ($C_{p_1}^i$ and $C_{p_2}^i$ are partial costs for node S_i)
Result: $C(Q(n), A(n))$
$C_{p_1}^1 \leftarrow x_1 T_s^1 + y_1 T_c^1$, $C_{p_2}^1 \leftarrow z_1 T_t^1$
for $i \leftarrow 2$ to N **do**
  $C_{p_1}^i \leftarrow x_i T_s^i + y_i T_c^i + \mid \min(x_i T_s^i + y_i T_c^i + z_i T_t^i - C_{p_1}^{i-1}, 0) \mid$
  $C_{p_2}^i \leftarrow z_i T_t^i + C_{p_2}^{i-1}$
  $C(Q(n), A(n)) \leftarrow C_{p_1}^i + C_{p_2}^i$
end for

Taken the previous definitions, a star-plus topology for $N > 1$ can be written in a similar way.

5.3.2.4 Solution

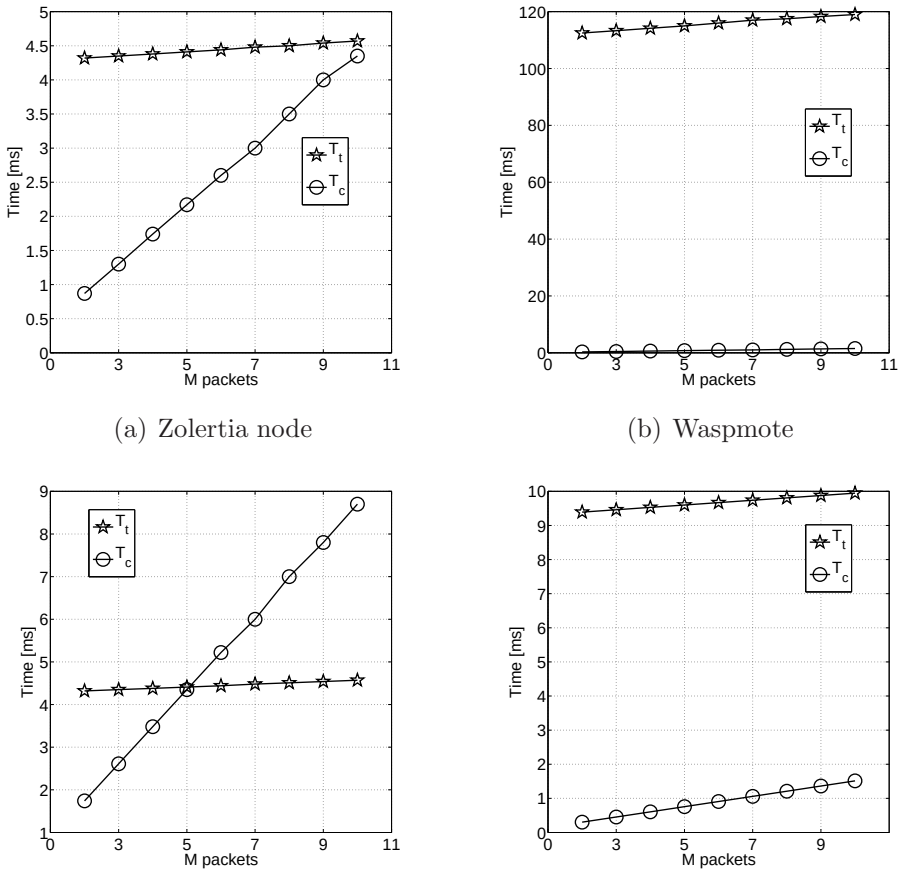
We apply the well-known value iteration algorithm [119] to solve the appropriate policies for our schemes. Our main goal, total energy consumption,

Algorithm 5 Value iteration algorithm for computing the total energy consumption

Data: $\mathcal{P}$, C, A, Q, Θ, η, ζ
Result: E_{total}
$V_0 \leftarrow O, j \leftarrow 0$
repeat
  $j \leftarrow j + 1$
  for all states Q(n)$\in \Theta$ **do**
    $V_j(Q(n)) \leftarrow \min_{A(n)}(C(Q(n), A(n)) + \eta \sum_{Q(n+1)\in\Theta} \mathcal{P}_{(Q(n),Q(n+1))}(A(n))V_{j-1}(Q(n)))$
  end for
until $\max_{Q(n)}(V_j(Q(n)) - V_{j-1}(Q(n))) < \zeta$
$E_{total} = V_j(1)$

is given by Algorithm (5), where the input data includes the state matrix Q, state set Θ, transition probability matrix $\mathcal{P}$ ($\Theta \times \Theta \times A$), cost matrix C ($\Theta \times A$), action matrix A ($\Theta \times N$), discount factor η, and value iteration ζ. The output of the algorithm V ($\Theta \times 1$) represents the expected objective value obtained following the policy from each state Q as defined in the Bellman equations. Since we want the energy consumption from the first state of the model, our interest metric is equivalent to the first value of V, meaning the value of the first state after j iterations, namely $V_j(1)$.

The parameters $\alpha, \beta, \gamma, \epsilon$, and P_T are introduced to the cost matrix C. For instance, for one simple node and *no feedback* scheme, the cost $C(Q(n), A(n))$ is given by (a) $\epsilon_i T_s^i P_t^i$, if $a_i(n) = s$, (b) $\beta_i T_c^i P_t^i$, if $a_i(n) = c$, (c) $(1 + \beta_i)T_t^i P_t^i$, if $a_i(n) = t$. The cost for listening and receiving feedback are added at the end when only one acknowledgement of the complete transmission of the packets is sent by the BS, such as $[(\alpha_i + \beta_i)T_r^i + \gamma_i\alpha_i T_l^i)]P_t^i$. In case of *full feedback* and $a_i(n) = t$, the energy budget is given by $[(1 + \beta_i)T_t^i + (\alpha_i + \beta_i)T_r^i + \gamma_i\alpha_i T_l^i]P_t^i$. For *min feedback*, the cost is $C(Q(n), A(n)) = [x_i\epsilon_i T_s^i + y_i\beta_i T_c^i + z_i((1+\beta_i)T_t^i + (\alpha_i+\beta_i)T_r^i + \gamma_i\alpha_i T_l^i)]P_t^i$. Thus, one difference between *full feedback* and *min feedback* is that in the first case the cost for feedback is calculated after each transmission, while for *min feedback* the cost for feedback is added after the completion of a set of actions, which may contain one or more transmissions. In case of *full feedback*, the number of transmissions is expected to be higher than the one for coding, since an unsuccessfully transmitted packet is not discarded, but retransmitted. For *no feedback* the number of operations for coding and transmission is similar as we discard each packet that is successful or not transmitted.



(a) Zolertia node

(b) Waspmote

(c) TelosB

(d) Mica

Figure 5.5: Time to perform one coded packet (T_c) on $\mathbf{GF}(2^8)$ out of M packets, each packet of 114 bytes and to transmit the coded packet (T_t) for different sensor hardware.

5.4 Numerical Results

We present the numerical results into two categories. We optimize the protocol to match the hardware, then the protocol and the hardware jointly. Our protocols are oriented to different types of feedback, maintaining the transmitted data in the same class (e.g., coded data). They are designed for a specific erasure probability, size, and type of data packets. The idea is not to show the benefits of network coding, but to understand the impact it has on the total energy consumption of the network.

Nodes are assumed to use the 802.15.4 protocol [109] for communication, which considers that the PHY packet has a payload of at most 127 bytes with

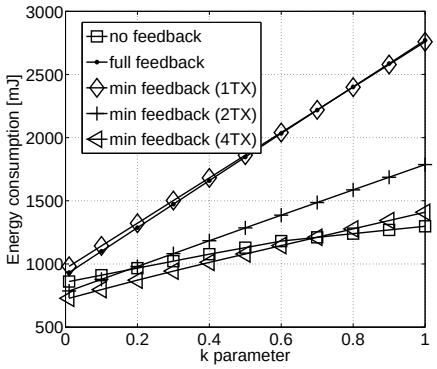
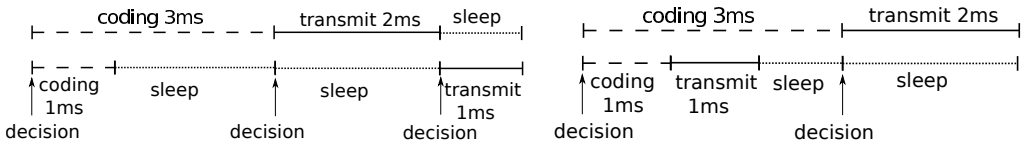


Figure 5.6: The variation of parameter k defined for the time to listen to an acknowledgement packet/packet, for star-plus topology, one node W and one relay W, $L = 114$ bytes, $Ack = 11$ bytes, $\mathbf{GF}(2^8)$, $M = 10$ packets, $Y = 4$ (for $no\ feedback$ and $full\ feedback$ scheme), erasure probability $e_i = 0.8$, $e_{i_R} = 0.1$ and $e_{R_i} = 0.1$.

6 additional bytes used for the protocol's header, whereas the MAC packet size is 114 bytes plus 13 for the overhead (short addressing and unsecured frame). The acknowledgement consists of 11 bytes.
The results are performed here for 4 types of commercial sensors. Zolertia is characterized by a CC2420 radio and a 16-bit Texas Instruments μC, Waspmote with a XBee radio and an 8-bit ATmega μC, TelosB with the same radio as Zolertia and a 16-bit Texas Instruments μC, different than the one for Zolertia, and Mica with a TR1000 radio and an 8-bit ATmega μC, different than the one for Waspmote. For simplicity, we identify the various sensors by their initials, i.e., Z for Zolertia, W for Waspmote, T for TelosB, and Mi for Mica. The parameters α, β, γ, and ϵ are computed according to the technical specifications for each platform. We average the value for the number of cycles Cyc using the related reference for Atmel [110], Texas Instruments [111] and [120]. The switching time between active and sleep modes according with the specifications is very small. Moreover, Z has T_t^i for a packet higher than the one to code a packet, but they become closer when the number of packets is increased, as shown in Figure 5.5. Moreover, W shows that the difference between T_t^i and T_c^i is much more higher than in the case of Z. We choose intentionally a sensor with such a high energy consumption to show the impact it has on the overall energy consumption system. Then, a T mote demonstrates that T_t^i can be higher or lower than T_c^i. Finally, Mi has T_t^i higher than T_c^i, but not so high as for the W case.
In case of $min\ feedback$ scheme, 6 joint actions are chosen for a node S_i: 1. $(x_i = Y,\ y_i = 0, z_i = 0)$; 2. $(x_i = Y, y_i = Y, z_i = Y)$; 3. $(x_i = Y, y_i =$



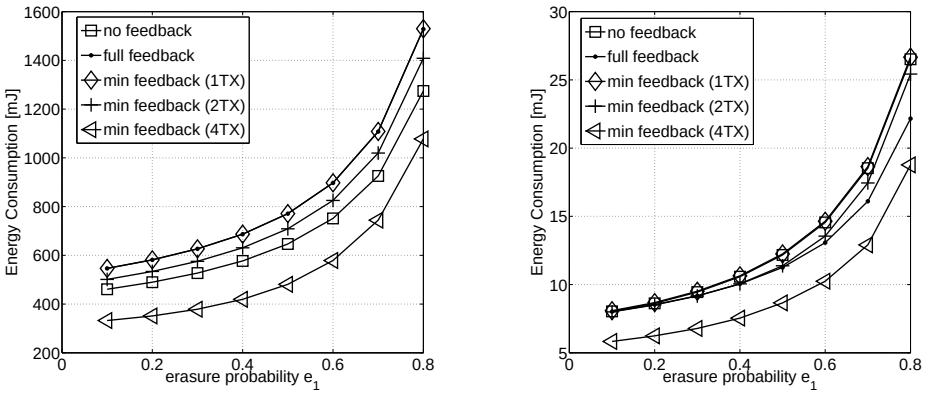
(a) *full feedback* and *no feedback* (b) *min feedback*

Figure 5.7: Small example of the cost calculation and decision model for three communication protocols.

$0, z_i = Y$); 4. $(x_i = Y, y_i = Y, z_i = 0)$; 5. $(x_i = 1, y_i = 0, z_i = 1)$; 6. $(x_i = 1, y_i = 1, z_i = 1)$. For a star topology with two nodes S_i and S_j, node S_j performs actions by taking into consideration the joint possibilities from Table 5.2. For a star-plus topology, the relay node chooses actions according to Table 5.1 (e.g., for set 4, the possibilities for the relay are either sleeping, coding or transmission). Also, we consider the *min feedback* scheme after each transmission ($1TX$) if $Y = 1$, after 2 transmissions ($2TX$) when $Y = 2$, and after 4 transmissions ($4TX$) when $Y = 4$. The value of Y does not influence the results for a star topology and *no feedback* or *full feedback*, because in a *no feedback* scheme a packet is discarded from the buffer after each transmission, while for *full feedback* the same packet is retransmitted until is successfully received at the BS.

In the value iteration algorithm, the parameters are fixed, i.e., $\eta = 1$ and $\zeta = 0.001$. Also, T_l^i for an acknowledgement or data packet is defined here by $T_l^i = k \cdot T_t^i$, where $0 < k \leq 1$. The variation of k for different schemes is provided in Figure 5.6 for a node W and a relay W. The parameter k needs to be carefully chosen according to both the protocol communication and the sensor platform. Moreover, the time to receive an acknowledgement packet is different than the time to receive a data packet, as $L = 114$ bytes and $Ack = 11$ bytes.

Before giving the numerical results, we provide a simple example. We consider the case of $M = 1$, $N = 2$, $Y = 1$, $T_c^1 = 3$ ms, $T_t^1 = 2$ ms, $T_c^2 = 1$ ms, $T_t^2 = 1$ ms, and $e_1 = e_2 = 0$, illustrated also in Figure 5.7. Without adding the feedback costs, the *full feedback* and the *no feedback* protocol need the following time slots to transmit one coded packet from each node to the BS: (1) both nodes are coding, thus, the required time is $C(Q(1), A(1)) = \max(T_c^1, T_c^2) = 3$ ms, (2) node 1 is transmitting and node 2 is sleeping, given $C(Q(2), A(2)) = T_t^1 = 2$ ms, (3) node 1 is sleeping and node 2 is transmitting, thus, $C(Q(3), A(3)) = T_t^2 = 1$ ms, concluding with a completion time of 6 ms. For the *min feedback* protocol, we have: (1) node 1 is coding $(x_1 = 0, y_1 = 1, z_1 = 0)$, while node 2 is coding and, then, transmitting $(x_2 = 0, y_2 = 1, z_2 = 1)$, thus, $C(Q(1), A(1)) = max(T_c^1, T_c^2 + T_t^2) = 3$



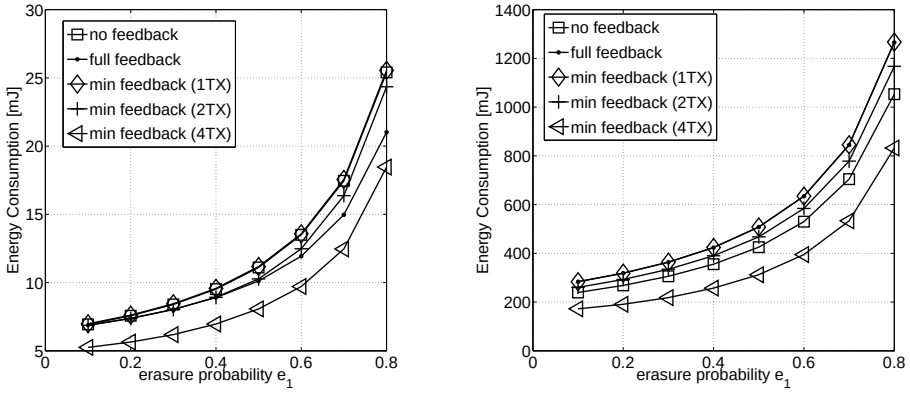
(a) two nodes W (b) two nodes Z

Figure 5.8: Energy consumption for two nodes in star topology, homogeneous hardware, $L = 114$ bytes, $Ack = 11$ bytes, $\mathbf{GF}(2^8)$, $M = 10$ packets, $k = 0.01$, erasure probability e_1 is varied, $e_2 = 0.1$.

ms, (2) node 1 is transmitting ($x_1 = 0, y_1 = 0, z_1 = 1$), while node 2 is sleeping ($x_2 = 1, y_2 = 0, z_2 = 0$), giving $C(Q(2), A(2)) = T_t^1 = 2$ ms, and the total completion time is now 5 ms. For *no feedback* scheme, the acknowledgement packet is sent only at the end of transmission. Both *full feedback* and *min feedback* ($1TX$) mean that the BS sends a feedback after a transmission occurs. But, for *full feedback* the first acknowledgement is after the second slot at 5 ms, while for *min feedback* ($1TX$) the first one is after the first combined action when the completion time is 3 ms, because the transmission of one node can start while the other node is still coding.
The numerical results provide examples for different instances, e.g, varying the number of packets or the erasure probability, where the metric is the energy consumption or the gain. We derive the gain from the energy and define it by comparing the energy consumption for a single node in a star topology with the energy consumed by a node in a star-plus topology. The nodes are assumed to be in the same transmission range. The plots are given for the most relevant combinations between the mentioned platforms.

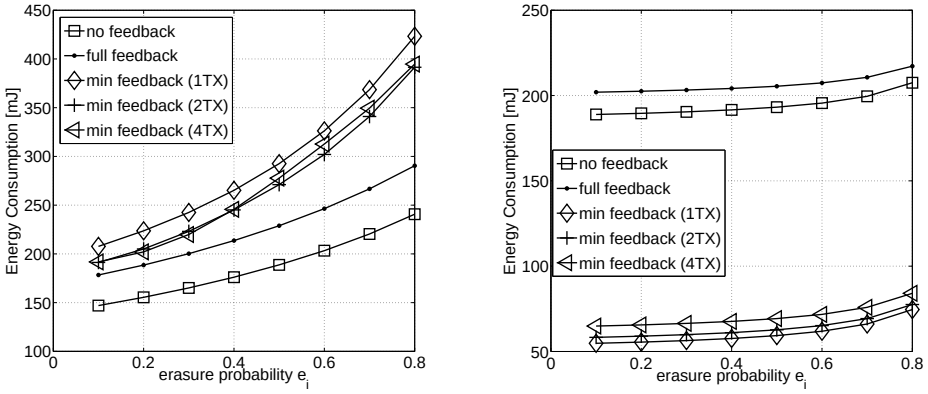
5.4.1 Optimizing the Protocol to Match the Hardware

Our first focus is directed to the various types of feedback used, from a minimal feedback, to intermediate feedback and to a per-packet basis. The findings are given in terms of energy consumption when the erasure between one source node and BS is varied, for various platforms for the source and



(a) one node Z and one node T (b) one node W and one node T

Figure 5.9: Energy consumption for two nodes in star topology, heterogeneous hardware, $L = 114$ bytes, $Ack = 11$ bytes, $\mathbf{GF}(2^8)$, $M = 10$ packets, $k = 0.01$, erasure probability e_1 is varied, $e_2 = 0.1$.



(a) node W, relay Z (b) node Z, relay W

Figure 5.10: Energy consumption for one node in star-plus topology, $L = 114$ bytes, $Ack = 11$ bytes, $\mathbf{GF}(2^8)$, $M = 10$ packets, $Y = 4$ (for *no feedback* and *full feedback* scheme), $k = 0.01$, erasure probability e_i is varied, $e_{i_R} = 0.1$ and $e_{R_i} = 0.1$, for various platforms.

relay node. Then, we show the analysis for the variation of the erasure probability of the different channels.

Feedback Analysis

We first pick two homogeneous nodes in a star topology network, as shown in Figure 5.8. The findings prove that the *min feedback* ($1TX$) scheme gives the worst performance and *min feedback* ($4TX$) the best one for both cases, the one with two nodes W and the one with two nodes Z. The gap between these two minimalistic feedback protocols varies between 0.6 and 0.7. Moreover, the simple schemes with *full feedback* and *no feedback* show different findings depending on the hardware used. For example, *full feedback* performs worst than *no feedback* for two nodes W and better than *no feedback* for two nodes Z. This happens because the listen and receive energy for node W ($\gamma = 1$, $\alpha = 2.28$) is much higher than the one for Z ($\gamma = 0.022$, $\alpha = 1.08$).
Second, we choose two heterogeneous nodes, as in Figure 5.9. Similar situation happens here, where the gap between the energy performance for *min feedback* ($1TX$) and *min feedback* ($4TX$) is again between 0.6 and 0.7, while the performance for the other protocols is according to the sensor hardware. Also, we can observe that by comparing Figure 5.8(a) to Figure 5.9(b), a similar behaviour occurs, with a bit lower energy consumption for Figure 5.9(b), because a node W has the dominant energy budget among the other used platforms.
Third, we switch to a star-plus topology with one node and one relay of different platforms, as it is shown in Figure 5.10. Here, the dependence of each protocol on the used hardware is highly observed. For instance, *min feedback* ($1TX$) acknowledges to be the worst choice for the combination node W and relay Z, with the energy variation between 208 mJ and 423 mJ, and the best one for node Z and relay W, where the energy consumption is between 54 mJ and 75 mJ. Also, *min feedback* ($4TX$) performs close to *min feedback* ($2TX$) for node W and relay Z, and worst than *min feedback* ($2TX$) for a node Z and a relay W. Also, the findings reveal that a W relay is not a good choice for *full feedback* protocol, because, as mentioned before, this platform has a high receive and listen energy. Moreover, when the erasure probability between the node Z and the BS increases, the *no feedback* scheme approaches the *full feedback* performance, because the relay W is used more for transmission, which implies also more energy consumption for listen and receive.
Comparing the results without relay with the ones with relay, we extract the following conclusions. (1) Two nodes W can spend up to 1600 mJ, as shown in the scenario from Figure 5.8(a). If the same node W is supported by a relay node Z, then the total energy consumption is almost up to 450 mJ, as Figure 5.10(a) demonstrates. This means that for two nodes W with two relays Z the energy can be reduced to at least 900 mJ. (2) The same situation

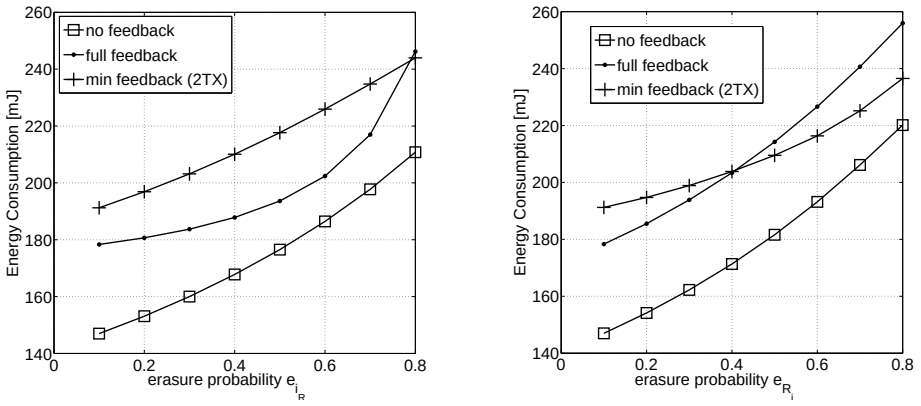
is not valid for two nodes Z in a star topology and one node Z and one relay W in a star-plus topology, as Figure 5.10(b) shows much worst results than Figure 5.8(b).
Feedback analysis concludes that (1) for two nodes of the same or not hardware in a star topology consume less energy if the protocol uses multiple actions with minimal feedback, such as *min feedback* ($4TX$), (2) while for a star-plus topology, the use of the relay may or not reduce the energy consumption and the choice of the protocol is dependent on the hardware.

Erasure Probability

In the following, we show the numerical results for a star-plus topology when the erasure probability is varied. The findings are given in Figure 5.11, where the energy consumption for a node W and a relay Z is exemplified when one of the erasure probability e_{i_R} or e_{R_i} is varied, while the other two are fixed. For simplicity, we limited the evaluation only for 3 schemes, i.e., *no feedback*, *full feedback*, and *min feedback* ($2TX$). The findings show that by varying any erasure probability, *no feedback* gives the best performance for the chosen platforms. This is feasible for a node W with a high energy consumption for listen and receive, since this protocol requires no acknowledgement between the transmissions. For the other two schemes, the results demonstrate the dependence on either e_{i_R} or e_{R_i}. For instance, the *full feedback* outperforms *min feedback* ($2TX$) for any value of e_{i_R} and when $e_{R_i} < 0.4$. We conclude that *no feedback* for a star-plus topology scheme shows the highest energy efficiency for any percentage of losses in the communication channel, when the node has the energy budget for listen end receive higher than the one for the relay.

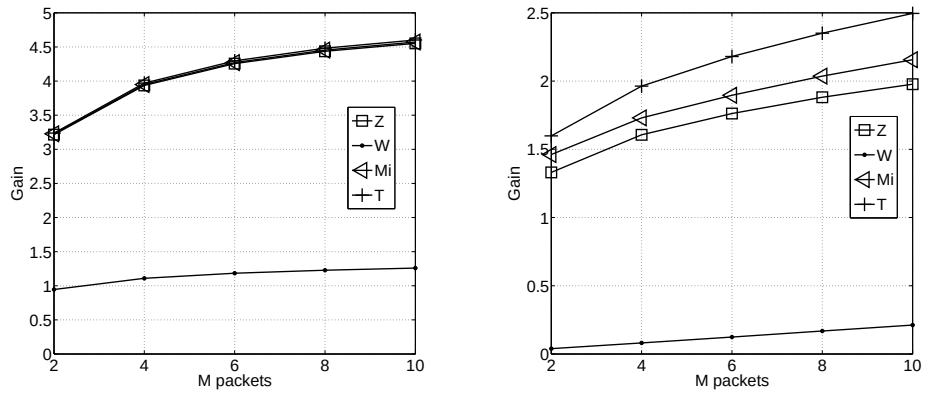
5.4.1.1 Optimizing the Protocol and the Hardware Jointly

We select a protocol and choose various platforms for the relay, while maintaining the same source hardware. Our main goal is to find the best hardware combination for the sensor and relay given a specific protocol. We show the results for one scheme, but the experiment can be repeated in the same manner for the other protocols. The findings are given in terms of gain in Figure 5.12. First, we choose *no feedback* scheme, a platform W for the source node, and vary the hardware of the relay node. Here, by selecting any of the platforms T, Mi or Z for relay, we get the same reduction in the energy budget, as Figure 5.12(a) proves. Therefore, the energy consumption can be reduced between 3.22 and 4.55 times depending on the number of packets. Less improvement is obtained for a relay W, where the gain is only between



(a) e_{i_R} is varied, $e_i = 0.1$ and $e_{R_i} = 0.1$ (b) e_{R_i} is varied, $e_i = 0.1$ and $e_{i_R} = 0.1$

Figure 5.11: Energy consumption for one node in star-plus topology (a node W and relay Z), $L = 114$ bytes, $Ack = 11$ bytes, $\mathbf{GF}(2^8)$, $M = 10$ packets, $Y = 2$ (*no feedback* and *full feedback* scheme), $k = 0.01$, varying the erasure probability.



(a) source node W, relay platform is varied (b) source node Z, relay platform is varied

Figure 5.12: Gain for one node in star topology vs one node in star-plus topology: $L = 114$ bytes, $Ack = 11$ bytes, $\mathbf{GF}(2^8)$, $Y = 2$ (for *no feedback* and *full feedback* scheme), $k = 0.01$, erasure probability $e_i = 0.8$, $e_{i_R} = 0.1$ and $e_{R_i} = 0.1$, *no feedback* scheme, varying the number of packets.

0.94 and 1.25. Second, we repeat the experiment when the source node is Z. Here, the best choice for the relay is a platform T with an improvement for the energy consumption up to 2.5 times, as Figure 5.12(b) demonstrates. In this case, the findings for a source Z also show different energy gains when choosing the hardware for relays among T, Mi, Z or W, with a significantly

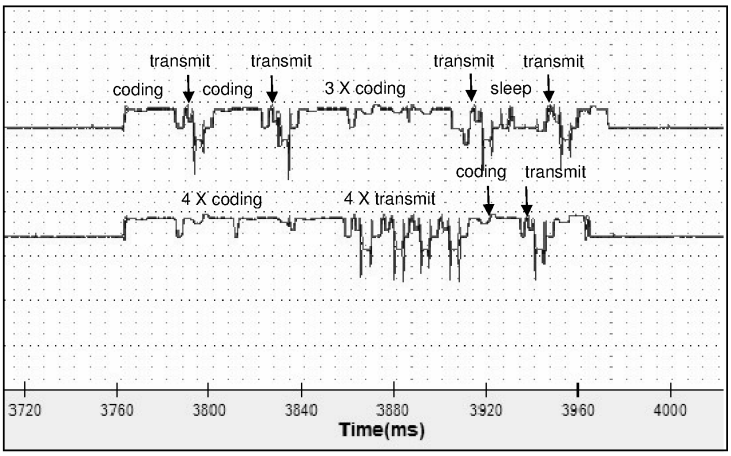


Figure 5.13: Sequences of performing simultaneously actions for two nodes using Power Monitor device.

decrease for a relay W. In conclusion, the combination node W with any relay among T, Mi or Z confirms the highest gain in terms of energy consumption, where the network's energy consumption decreases up to 4.55 times than in the case of a simple node of type W.

5.5 Implementation Results

We implement our protocols in real-life measurements by using TelosB motes (UC Berkeley, Crossbow) running TinyOS 2.1.1 operating system. Measurements are performed by using the TinyOS timer module that can track the time a sensor spends while performing a specific operation. We also use the Power Monitor device provided by the Monsoon Solutions Inc. [112] for a better accuracy of the measured values.

5.5.1 Protocol Measurements

For the implementation, we use three motes in a star topology, one as a BS and two as sensor nodes. The steps are explained in the following and are valid for any of the three protocols proposed in Section III. The erasure channel between the sensors and BS is deployed depending on the distance between the sensors and BS. We fixed the erasure probability between one sensor and BS and vary the erasure probability between the other sensor and BS. Thus, we found cases where the erasure varies between 0.05 and 0.71 for the measured results and we choose between 0.1 and 0.8 for the theoretical results, as Figure 5.14 shows.

- **Define the action sets**: First, we define the sets of optimum actions for each node, as described in Section III. We apply our MDP and use as input parameters the active periods for different energy profiles previously measured. These sets contain actions that the two nodes can perform simultaneously in order to minimize the energy efficiency of the network.

- **Characterize the state model**: We define a state model that includes a list with the actions used by the transmitting nodes, e.g., *code*, *transmit*, *sleep*, *listen*, and BS, e.g., *initialization*, *synchronize*, $feedback$, *sleep*. The states for the nodes are simply described by their names. BS starts the process by sending to each sensor the list with the actions and the corresponding time intervals they need to carry out. This coincides to the *initialization* state. Then, in the *synchronize* state, BS sends a broadcast message for the initiation of the transmission process and the nodes start the process only after the reception of this message. The order and the time of executing the actions by each sensor is given by the action sets previously defined. The feedback is also introduced depending on the protocol. For *no* $feedback$, we simply define a last listening state for each sensor, for $full\ feedback$, the listening state is defined after each *transmit* state, where for *min* $feedback$ is given after a group of actions, as described by the protocol. Figure 5.13 provides an example using the Power Monitor device of the concurrent actions performed by the two nodes in a star topology.

- **Validation of the results**: We program the sensors using the state model and cross-validate the results. We check if the proposed model fulfils the requirements of the real sensors. We run 3500 times each protocol and obtain a successful completion of the process with a percentage between $98.6\% - 98.9\%$. This percentage is calculated based on (1) successful completion of the actions (coding, transmission, etc.) within the time intervals we defined, (2) interference produced by simultaneously transmissions (here, we enable the carrier sense-multiple access protocol, as defined by TinyOS in CC2420 radio stack), and (3) the time deviation between the starting point of the transmission process of the nodes, which includes the correct reception of each packet at BS.

The results from the implementation are compared with the theoretical results for two TelosB nodes in a star topology and represent (I) measured results, including (a) the total energy without any decision model, just as described in equation (4.1), and including just one acknowledgement about

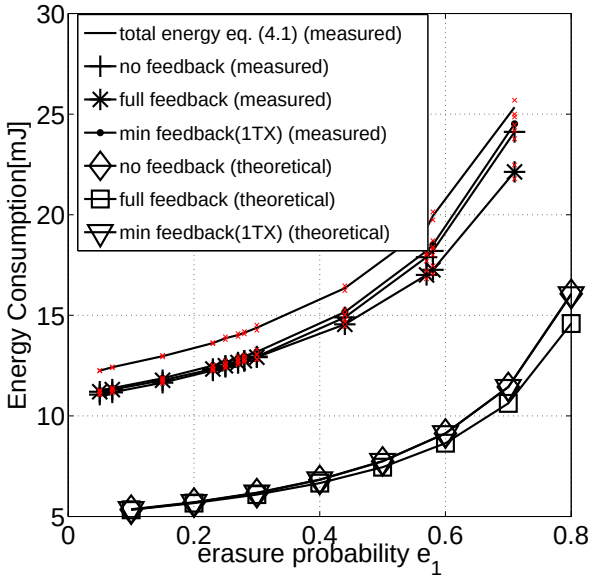


Figure 5.14: Energy consumption for $N = 2$ nodes in a star topology, $M = 10$ packets, $L = 100$ bytes, $Ack = 11$ bytes, $\mathbf{GF}(2^8)$, erasure probability e_1 is varied, $e_2 = 0.1$.

the reception of all the packets by BS, (b) the total energy using the decision model for all three proposed schemes, and (II) theoretical results, including the total energy using the decision model for all three proposed schemes. The plots are shown in Figure 5.14. The theoretical results use the model described in Section III B-1) for the coding operations and the data specification of the sensor [76], [88], [102]. Since this model is described for a dedicated hardware and the parameters of the sensor are chosen at their optimum values, as explained in the sensor's datasheet, the corresponding data gives a lower bound of the energy level. For instance, the transmission data rate is set at 250 kbps in the sensor's data specification, but in practice this value can hardly be reached. We also add 19 bytes overhead to the data payload (13 bytes for the MAC layer and 6 bytes for the PHY layer, as described in the IEEE 802.15.4). Furthermore, the measured results are based on software description model and are obtained using the timer module from TinyOS 2.1.1 and the Power Monitor device from Monsoon Solution Inc.. The results represent an average of 3500 runs and 97.5% confidence intervals. Here, the transmission time is the summation of the T_{SPI} and T_t, because T_{SPI} is non-negligible. The upper bound is given by the total energy consumption model without any decision, as explained in equation (4.1). For the decision model, additional initializations and synchronization cost oper-

ations are added, i.e., the BS computes the algorithm based on the MDP and sends a message to each node indicating the actions and time intervals for each sensor, and also, sends a synchronization message for the beginning of the communication process by the nodes. Since BS is assumed not to be battery powered, we only count the energy consumed by the nodes when receiving the initialization messages from BS, which is 1.23 mJ. TelosB sensor nodes are characterized by high transmit energy comparing to the processing energy ($\beta = 0.1$). However, the results for the total energy model using equation (4.1) can be improved with the decision model, as Figure 5.14 illustrates. Moreover, the listen energy is low ($\gamma = 0.022$) and the receive energy is approximately equal to the transmit energy ($\alpha = 1.08$), which means the cost related to the feedback used does not influence significantly the total energy, but the decision to discard or not the packets does. This concludes with less energy consumption for the $full\ feedback$ scheme.

5.6 Concluding Remarks

Understanding the energy consumption and the design of communication protocols in WSN requires an accurate yet simple hardware abstraction with a total energy model that is calculated based on the transmit, receive, processing, idle/listen, sleeping, and switching energy consumption. Our results showed the need for this model to the dependence of the communication efficiency on the energy profile of the underlying hardware. Consequently, protocol optimization is based on the proposed methodology for feedback techniques that allow the designer to choose the less costly communication protocol, given the hardware description of each sensor and the challenges in the communication requirements. For single hop transmissions with the same or different hardware, a protocol with minimalistic feedback performs the best, whereas for networks with relays, the energy characteristics of the sensor nodes decide the performance. Moreover, we showed matches between the hardware of the sensor and the communication protocol that decrease the total consumption of the system 4.5 times. We also cross-validated the results using real-life measurements.

Chapter 6

Conclusions and Future Work

Motivated by the applications where the delay and the energy consumption are critical, we focused on the design optimization of the communication protocols using network coding. The solutions provided by network coding seem to be well-known from the throughput perspective, but not so well from the delay perspective. With this in mind, we analysed the scenarios where the transmission delay is critical, e.g., real time applications, and provided the delay distribution in one-to-many scenarios. Then, we evaluated the transmission delay for the data gathering applications and challenged networks in many-to-one networks. Furthermore, we understood that the data transmission delay is directly connected to the design of energy efficient protocols. Thus, we modelled the total energy consumption of the system based on the hardware descriptions of the platforms and offered optimized solutions for the design of the communication protocol. We now have an answer to the question from the introduction. Hence, we are able to minimize the transmission delay and to design energy-efficient protocols by carefully matching the requirements of the application and including the prerequisites imposed by the network and the application design.

We present the main original contributions of the thesis and the possible directions for the future research work.

Analysis of the Delay Distribution for Broadcast Applications

In Chapter 2, we explored the delay behaviour of network coding in the case of finite field sizes and an arbitrary number of data packets. The first analysis was based on a closed-form expression. This is valid for one receiver, but gives crucial insights of the delay distribution. More precisely, we showed that, even for a small finite field size, the probability of having M linearly independent combinations after M received symbols is already close to 1. We

also demonstrated that to obtain a similar performance to the standard ARQ scheme, one can use RLNC in a finite field of small size, without the need of a feedback channel. We extended this analysis to two receivers for erasure broadcast channels by introducing a Markov chain model. Our findings were compared with the well-known scheme, such as ARQ with perfect feedback, round robin scheduling and a class of fountain codes. The comparison reveals that network coding on $\mathbf{GF}(2^4)$ offers the best delay performance for two receivers and $\mathbf{GF}(2)$ induces a heavy tail in the delay distribution, which implies that network coding based on XOR operations bears a relevant cost in terms of worst-case delay. For the case of more receivers, which is mathematically challenging because requires taking complex statistical dependencies into consideration, we proposed a brute-force methodology. The model shows the delay distribution of network coding for small generations and field size up to $\mathbf{GF}(2^4)$. The idea behind this method is to fix the pattern of packet erasures and to try out all possible encodings for various system and channel parameters. Our findings can be used to optimize network coding protocols with respect not only to their average but also to their worst-case delay.

One important remark here is that the proposed brute-force methodology to compute the delay distribution can be used not only for RLNC but also for other coding schemes. Uncoded packets are an example, whose performance we compared against the case with coding. It would be interesting to understand if the delay for random network coding with some constraints (for example, in the choice of coefficients) is still well approximated by a normal distribution.

We faced a high-dimensional and computationally demanding problem in computing the delay distribution. What we proposed for further analysis is a combination of the methods, the brute-force and the Markov chain. In particular, a hybrid search would be able to select the most relevant erasure patterns while capitalizing on the Markov chain to take the impact of the field size into account. Devising such strategies is part of the future work.

Data Collection Protocol using Tunable Sparse Codes and Feedback Mechanisms

In Chapter 3, the data transmission time has been evaluated for the data gathering applications, using inter-flow network coding. Inspired in part by the previous work in [36] and aimed to build a robust network coding protocol for challenged networks, we analysed the performance of TSNC and

RLNC. Hence, we applied various sparsity levels of the coded packets and different feedback types. First, we proposed a mathematical formulation for a line network that models the flow of innovative/non-innovative packets. The analytical bounds are valid for any field size and apply for the pure RLNC, the sparsity, the explicit feedback, and the implicit feedback case. Second, since the previous results are valid only for a line network, we extended the analysis to a grid network. Here, we performed numerical results using a small field size and compared the findings with a master slave protocol, conventional RLNC and with some tuned versions of RLNC. The results demonstrated that the network topology dictates the protocol performance and the completion time increases with the sparsity. Moreover, our results show that the minimum completion time is achieved either by using very sparse coded packets and frequent feedback or dense coded packets and a less frequent feedback. The most important insight is that minimizing delay performance can be achieved for a wide variety of sparsity-feedback pairs allowing network designers to choose a setting that best matches their devices capabilities.

The proposed methodologies can be extended to any number of nodes in the network, where the mathematical model for the line network is valid for any field size, and the numerical performance for the grid network applies only for $\mathbf{GF}(2)$. Designing a mathematical model for a general network topology is part of our future work.

Hardware Abstraction Model for Protocol Design

Chapter 4 provided an analysis of the relevant hardware characteristics of various platforms showing that the transmission energy is not the main source of energy consumption for the majority of sensor platforms. This was clearly an evidence that for an accurate characterization of the energy consumption, a total energy model is needed. With this in mind, we built a simple hardware abstraction model that takes into consideration the transmission, processing, receiving, listening, sleeping, and switching energy profiles. Moreover, we illustrated the model by evaluating the energy cost of two communication protocols under two different hardware assumptions. Thus, we considered the microcontroller characteristics and a (i) hardware and (ii) software description of the arithmetic operations. The findings show that (a) the energy cost for a protocol varies significantly on different sensor platforms and (b) the protocols can be adapted to the underlying hardware for a maximum energy efficiency. Hence, given the importance of the processing of the data, we selected the hardware and switched between protocols with and without

coding. As a result, we found cases in which coding brings benefits in terms of energy consumption, and other cases, where with network coding, the performance decreases. Our results indicate that, by leveraging our proposed hardware abstractions, energy efficiency can be significantly improved either by (1) developing hardware-adaptive protocols, or (2) choosing standard protocols judiciously in order to match the underlying hardware.

The findings for (ii) were supported with real-life measurements using TelosB motes. More specifically, we measured the active periods of time using TelosB motes for coding and transmitting coded/uncoded packets, under different packet and field size. Hence, for the data transmission time, we demonstrated that other factors, such as the SPI time, contribute to the energy performance of the system. Moreover, an important insight for the future optimization network coding protocols is that using a field size multiple of μCs data registers is less energy consuming. This happens because the conversion of the packets into symbols is straightforward. For the case of a field size not multiple of μC data register, e.g., $\mathbf{GF}(2^4)$, the implementation requires two more functions, one to convert the each packet into symbols and the other is to convert back the combinations obtained from RLNC operations into coded packets. These conversion functions definitely decrease the system performance in terms of energy consumption, as shown in this chapter. These results are opposite from the theoretical results for the delay distribution, where the field size $\mathbf{GF}(2^4)$ is high enough to achieve the best performance.

The next natural step is to extend the real-life measurements to other platforms and to understand the limitations of network coding on general battery-powered devices.

Protocol Optimization for Deadline Constrained Applications

With Chapter 5, we demonstrated the need in the communication efficiency of the hardware abstraction model provided in Chapter 4. Thus, we modelled the communication protocol from the point of view of the energy consumption using feedback based techniques and under the challenges imposed by the hardware description of the nodes, the application, and the network channel. We provided the less costly communication protocol using a MDP model for single and multiple hop transmissions. The results were divided into theoretical and measured, both parts given solutions for the future protocol optimizations. Thus, the theoretical findings show that for a single hop transmissions with the same or different hardware, a protocol with minimal-

istic feedback performs the best, whereas for multiple hop transmissions, the energy characteristics of the sensor nodes decide the performance. Moreover, we showed matches between the hardware of the sensor and the communication protocol that decrease the total consumption of the system 4.5 times. With the real-life measurements, we demonstrated that our protocols can be simply implemented. A comparison between the theoretical and measured results was also included.

The work shed light on the relation between the hardware of each sensor and the design of the communication protocol in WSN. The analysis we provided here let some open questions for the joint effect of the network topology and the hardware parameters for the overall energy budget.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

Chapter 7

Bibliography

[1] S. Sezer, S. Scott-Hayward, P.K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, pp. 36–43, July 2013.

[2] C. Wang, F. Haider, X. Gao, X. You, Y. Yang, D. Yuan, H. Aggoune, H. Haas, S. Fletcher, and E. Hepsaydir, "Cellular architecture and key technologies for 5g wireless communication networks," *IEEE Communications Magazine*, vol. 52, pp. 122–130, February 2014.

[3] M. Wen and V. Li, "Form follows function: Designing smart grid communication systems using a framework approach," *IEEE Power and Energy Magazine*, vol. 12, pp. 37–43, May 2014.

[4] H. Farhangi, "A road map to integration: Perspectives on smart grid development," *IEEE Power and Energy Magazine*, vol. 12, pp. 52–66, May 2014.

[5] R. Ahlswede, N. Cai, S.-Y.R. Li, and R.W. Yeung, "Network information flow," *IEEE Transaction Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[6] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transaction on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

[7] T. Ho, M. Médard, J. Shi, M. Effros, and D.R. Karger, "On randomized network coding," in *Proc. of 41st Annual Allerton Conference on Communication, Control and Computing*, October 2003, vol. 1, pp. 1–10.

[8] P.A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings 41st Allerton Conference Communication, Control and Computing*, Monticello, IL, October 2003.

[9] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transaction Information Theory*, vol. 52, no. 10, pp. 4413–4430, Octomber 2006.

[10] C. Fragouli, D. Katabi, A. Markopoulou, M. Médard, and H. Rahul, "Wireless network coding: Opportunities and challenges," in *IEEE Military Communication Conference*, Orlando, FL, USA, October 2007, pp. 1–8.

[11] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. of ACM Sigcomm*, New York,USA, 2007, pp. 169–180.

[12] J. Krigslund, J. Hansen, M. Hundebøll, D. Lucani, and F. Fitzek, "Core: Cope with more in wireless meshed networks," in *IEEE Vehicular Technology Conference*, Desden, Germany, June 2013, pp. 1–6.

[13] S. Katti, H. Rahul, W. Huss, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *ACM Sigcomm*, Pisa, Italy, September 2006, pp. 497–510.

[14] M. Pedersen, J. Heide, F. H. P. Fitzeka, and T. Larsen, "Network coding for mobile devices - systematic binary random rateless codes," in *Proc. ICC 09 - Workshop on Cooperative Mobile Networks*, Dresden, Germany, June 2009, pp. 1–6.

[15] M. Hundebøll, J. Ledet-Pedersen, J. Heide, M.V. Pedersen, S.A. Rein, and F.H.P. Fitzek, "Catwoman: Implementation and performance evaluation of ieee 802.11 based multi-hop networks using network coding," in *IEEE Vehicular Technology Conference*, Quebec City, Canada, September 2012, pp. 1–5.

[16] P. Vingelmann, F.H.P. Fitzek, M.V. Pedersen, J. Heide, and H. Charaf, "Synchronized multimedia streaming on the iphone platform with network coding," *IEEE Communications Magazine*, vol. 49, pp. 126–132, June 2011.

[17] J. Hansen, J. Krigslund, D. Lucani, and F. Fitzek, "Bridging inter-flow and intra-flow network coding for video applications: Testbed description and performance evaluation," in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*, Berlin, Germany, September 2013, pp. 7–12.

[18] Z. Li and B. Li, "Network coding in undirected networks," in *38th Annual Conference in Information Science and Systems*, Princeton, NJ, March 2004.

[19] P. Chaporkar and A. Proutiere, "Adaptive network coding and scheduling for maximizing throughput in wireless networks," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, New York, USA, 2007, pp. 135–146.

[20] A. Agarwal and M. Charikar, "On the advantage of network coding for improving network throughput," in *Information Theory Workshop*, Miami, October 2004, pp. 247–249.

[21] Z. Li, B. Li, D. Jiang, and L. C. Lau, "On achieving optimal throughput with network coding," in *Infocom*, Miami, March 2005, pp. 2184–2194.

[22] R.A. Costa, D. Munaretto, J. Widmer, and J. Barros, "Informed network coding for minimum decoding delay," in *Proc. of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 08)*, Atlanta, Georgia, September-October 2008, pp. 80–91.

[23] J. Barros, R. A. Costa, D. Munaretto, and J. Widmer, "Effective delay control in online network coding," in *Proc. IEEE Infocom*, Rio de Janeiro, Brazil, April 2009.

[24] D. E. Lucani, M. Stojanovic, and M. Médard, "Random linear network coding for time division duplexing: When to stop talking and start listening," in *IEEE Infocom*, Rio de Janeiro, Brazil, April 2009.

[25] A. Shokrollahi, "Raptor codes," *IEEE/ACM Transaction on Networking*, vol. 14, pp. 2551–2567, March 2006.

[26] T. K. Dikaliotis, A. G. Dimakis, T. Ho, and M. Effros, "On the delay of network coding over line networks," in *Proc. IEEE International Symposion on Information Theory*, Seoul, Korea, June 2009.

[27] J. K. Sundararajan, D. Shah, and M.Médard, "Feedback-based online network coding," *CoRR abs/0904.1730*, 2009.

[28] C. Fragouli, D. Lun, M. Médard, and P. Pakzad, "On feedback for network coding," in *Proc. 41st Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, USA, March 2007, pp. 248–252.

[29] L. Keller, E. Drinea, and C. Fragouli, "Online broadcasting with network coding," in *4th Workshop on Network Coding, Theory, and Applications*, Hong Kong, China, January 2008, pp. 1–6.

[30] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," in *IEEE International Symposion on Information Theory*, Toronto, Canada, July 2008.

[31] J. K. Sundararajan, D. Shah, and M. Médard, "Online network coding for optimal throughput and delay–the two-receiver case," *International Symposium on Information Theory and its Applications*, vol. 1, pp. 1–4, December 2008.

[32] R.A. Costa, D. Ferreira, and J. Barros, "Feber: Feedback based erasure recovery for real-time multicast over 802.11 networks," *CoRR abs/1109.1265*, 2011.

[33] R. Rout, S. Ghosh, and S. Chakrabarti, "Network coding-aware data aggregation for a distributed wireless sensor network," in *International Conference on Industrial and Information Systems*, Haikou, China, December 2009, pp. 32–36.

[34] R. Chandanala and R. Stoleru, "Network coding in duty-cycled sensor networks," in *Seventh International Conference on Networked Sensing Systems*, Kassel, Germany, June 2010, pp. 203–210.

[35] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli, "Sensecode: Network coding for reliable sensor networks," *ACM Transactions on Sensor Networks*, vol. 9, 2013.

[36] R. Prior, D.E. Lucani, Y. Phulpin, M. Nistor, and J. Barros, "Network coding protcols for smart grid communications," *IEEE Trans. on Smart Grids*, pp. 1–9, March 2014.

[37] Soheil Feizi, Daniel E. Lucani, and Muriel Médard, "Tunable sparse network coding," in *Proc. of the Int. Zurich Seminar on Comm.*, March 2012, pp. 107–110.

[38] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "A network coding approach to energy efficient broadcasting: from theory to practice," in *IEEE Infocom*, Barcelona, Spain, April 2006, pp. 1–11.

[39] T. Cui, L. Chen, and T. Ho, "Energy efficient opportunistic network coding for wireless networks," in *IEEE International Conference on Communications*, Glasgow, June 2007.

[40] X. Shi, M. Médard, and D.E. Lucani, "When both transmitting and receiving energies matter: An application of network coding in wireless body area networks," in *NC-Pro Workshop at IFIP Networking Conference*, Valencia, Spain, May 2011, pp. 119–128.

[41] S. Croce, F. Marcelloni, and M. Vecchi, "Reducing power consumption in wireless sensor networks using a novel approach to data aggregation," *Computer Journal*, vol. 51, no. 2, pp. 227–239, March 2008.

[42] Q. Wang, M. Hempstead, and W. Yang, "A realistic power consumption model for wireless sensor network devices," in *Sensor and Ad Hoc Communications and Networks*, Reston, VA, September 2006, pp. 286–295.

[43] R. Vidhyapriya and P.T. Vanathi, "Energy aware routing for wireless sensor networks," in *International Conference on Signal Processing, Communications and Networking*, Chennai, February 2007, pp. 445–450.

[44] M.J. Miller and N. Vaidya, "Minimizing energy consumption in sensor networks using a wakeup radio," in *Wireless Communications and Networking Conference*, Atlanta, Georgia, March 2004, pp. 2335–2340.

[45] F. Shebli, I. Dayoub, A.O. M'foubat, A. Rivenq, and J.M. Rouvaen, "Minimizing energy consumption within wireless sensors networks using optimal transmission range between nodes," in *IEEE International Conference on Signal Processing and Communications*, Dubai, United Arab Emirates, November 2007.

[46] A.G. Ruzzelli, P. Cotan, G.M.R. O'Hare, R. Tynan, and P.J.M. Havinga, "Protocol assessment issues in low duty cycle sensor networks: the switching energy," in *International Conference on Sensor Networks, Ubiquitos and Trustworthy Computing*, Taiwan, June 2006.

[47] R. Jurdak, A.G. Rizzelli, and G.M.P. O'Hare, "Radio sleep mode optimization in wireless sensor networks," *IEEE Transaction on Mobile Computing*, vol. 9, no. 7, pp. 955–967, July 2010.

[48] M. Liu, J. Cao, G. Chen, and X. Wang, "An energy-aware routing protocol in wireless sensor networks," *Sensors*, vol. 9, pp. 445–462, 2009.

[49] K. Akkaya and M. Younis, "Energy and qos aware routing for wireless sensor networks," *Cluster Computing*, vol. 8, no. 2-3, pp. 179–188, July 2005.

[50] K. Akkaya and M. Younis, "An energy-aware qos routing protocol for wireless sensor networks," in *International Conference on Distributed Computing Systems Workshops*, Providence, RI, USA, May 2003, pp. 445–450.

[51] *Micaz Datasheet*, Crossbow Technologies, 2006.

[52] *Waspmote Tech. Guide*, Libelium Comunicaciones Distribuidas S.L, 2010.

[53] D. E. Lucani, M. Stojanovic, and M. Médard, "Random linear network coding for time division duplexing: Field size considerations," in *Proc. Globecom*, Hawaii, USA, December 2009, pp. 1–6.

[54] D. E. Lucani, M. Stojanovic, and M. Médard, "Random linear network coding for time division duplexing: Energy analysis," in *Proc. ICC*, Dresden, Germany, 2009.

[55] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE Infocom*, Calcutta, India, December 2005.

[56] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network coding: An instant primer," *ACM Computer Communication Review*, vol. 36, pp. 63–68, January 2006.

[57] A. Eryilmaz, A. Ozdaglar, and M. Médard, "On delay performance gains from network coding," in *Annual Conf. on Information Sciences and Systems*, Princeton, NJ, March 2006, pp. 864–870.

[58] D. E. Lucani, M. Médard, and M. Stojanovic, "On coding for delay - new approaches based on network coding in networks with large

latency," *Proc. Information Theory and Application Workshop*, pp. 191–200, February 2009.

[59] M. Luby, "LT codes," in *Proc. 43rd Annual Symposion on Fundations of Computer Science*, Vancouver, Canada, November 2002, pp. 271–280.

[60] A. Mahmino, J. Lacan, and C. Fraboul, "Guaranteed packet delays with network coding," in *SECON Workshops '08*, San Francisco, CA, June 2008, pp. 1–6.

[61] E. Drinea, C. Fragouli, and L. Keller, "Delay with network coding and feedback," in *International Symposion on Information Theory*, Seoul, Korea, June 2009, pp. 844–848.

[62] D. J. C. Mackay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[63] L. Tassiulas, R. Tassiulas, and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transaction on Information Theory*, vol. 39, no. 2, pp. 466–478, March 1993.

[64] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2–16, 2003.

[65] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, November 2003, pp. 14–27.

[66] H. Lu, C.H. Foh, and J. Cai, "Scalable data dissemination protocol for wireless sensor networks," in *18th IEEE International Conference on Networks*, Singapore, December 2012, pp. 471–476.

[67] S. Pfletschinger, M. Navarro, and C. Ibars, "Energy-efficient data collection in wsn with network coding," in *IEEE GLOBECOM Workshops*, GLOBECOM Workshops, December 2011, pp. 394–398.

[68] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1948, December 1992.

[69] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *IEEE Transactions on Information Theory*, vol. 55, pp. 797–815, February 2009.

[70] S. Feizi, D.E. Lucani, C.W. Sõrensen, A. Makhdoumi, and M. Médard, "Tunable sparse network coding for multicast networks," in *accepted to the 2014 International Symposium on Network Coding*, June 2014.

[71] Simulink, "http://www.mathworks.com/products/simulink/," .

[72] M. Wang and B. Li, "How practical is network coding?," in *IEEE IWQoS*, New Haven, CT, US, June 2006, pp. 274–278.

[73] Y. Zhuang, J. Pan, and L. Cai, "Minimizing energy consumption with probabilistic distance models in wireless sensor networks," *IEEE INFOCOM*, pp. 1–9, March 2010.

[74] R. Srivastava and C.E. Koksal, "Basic tradeoffs for energy management in rechargeable sensor networks," *http://arxiv.org/abs/1009.0569v1*.

[75] L. Lin, N.B. Shroff, and R. Srikant, "Energy-aware routing in sensor networks: A large system approach," *Ad Hoc Networks*, vol. 5, no. 6, pp. 818–831, August 2007.

[76] *TelosB Mote Platform*, Crossbow Technologies, 2010.

[77] *Lotus Datasheet*, Memsic, 2011.

[78] *M16/62P User's Manual*, Renesas, 2006.

[79] *Zolertia Z1 WSN Datasheet*, Zolertia, 2010.

[80] *RFM1001 Datasheet*, RF Monolithics, 2008.

[81] *CC1000 Radio Transceiver Datasheet*, Texas Instruments, 2009.

[82] F. Jin, H.A. Choi, and S. Subramaniam, "Hardware-aware communication protocols in low energy wireless sensor networks," *IEEE Military Communications Conference*, vol. 1, pp. 676–681, October 2003.

[83] P. Sausen, J. Sousa, M. Spohn, A. Perkusich, and A. Lima, "Dynamic power management with scheduled switching modes," *International Symposion on Modelling, Analalysis, and Simulation of Computer and Telecommunication Systems*, pp. 1–8, November 2007.

[84] S. Jaggi, P A. Chou, and K. Jain., "Low complexity algebraic multicast network codes," *IEEE International Symposium on Information Theory*, July 2003.

[85] D. E. Lucani, M. Medard, and M. Stojanovic, "On coding for delay - network coding for time-division duplexing," *IEEE Transaction on Information Theory*, vol. 58, no. 4, pp. 2330–2348, April 2012.

[86] M. Nistor, D.E. Lucani, T.T.V. Vinhoza, R.A. Costa, and J. Barros, "On the delay distribution of random linear network coding," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1084–1093, May 2011.

[87] *RFM 1000 Datasheet*, RF Monolithics, 2008.

[88] *CC2420 Radio Transceiver Datasheet*, Texas Instruments, 2010.

[89] *TelosB CM5000 Datasheet*, Advantic, 2010.

[90] *Tmote Sky Datasheet*, Moteiv Corporation, 2006.

[91] *Imote 2 Datasheet*, Memsic, 2007.

[92] *Mica2(MPR400CB) Datasheet*, Crossbow Technologies, 2007.

[93] *Mica2(MPR300CA) Datasheet*, Crossbow Technologies, 2003.

[94] *Iris Datasheet*, Memsic, 2007.

[95] *Atmel AT86RF230 Datasheet*, Atmel Corporation, 2009.

[96] *Firefly Datasheet*, Carnegie Mellon University, 2010.

[97] *CC2520 Radio Transceiver Datasheet*, Texas Instruments, 2007.

[98] *VEmesh Product Specification*, Virtual Extension, 2010.

[99] *TinyNode 584 User's Manual*, Shockfish SA, 2011.

[100] *TinyNode 184 User's Manual*, Shockfish SA, 2010.

[101] *ATmega8535 Datasheet*, Atmel Corporation, 2005.

[102] *MSP430F1611 Datasheet*, Texas Instruments, 2011.

[103] *MSP430F149 Datasheet*, Texas Instruments Inc., 2004.

[104] *ATmega128L Datasheet*, Atmel Corporation, 2011.

[105] *MSP430F2618 Datasheet*, Texas Instruments Inc., 2011.

[106] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design and Test of Computers*, vol. 18, pp. 62–74, March 2001.

[107] L. Biard and D. Noguet, "Reed-solomon codes for low power communications," *Journal of Communication*, vol. 3, no. 2, pp. 13–21, April 2008.

[108] D.E. Lucani, M. Medard, and M. Stojanovic, "Systematic network coding for time-division duplexing," in *IEEE International Symposion on Information Theory*, Austin, TX, June 2010, pp. 2403–2407.

[109] *IEEE 802.15.4 Specification*, IEEE, September 2006.

[110] *Instruction Set Nomenclature*, Atmel, 2010.

[111] *MSP430X2XXX Family, User's Guide*, Texas Instruments, 2011.

[112] Monsoon Solutions Inc., "Mobile device power monitor manual ver 1.12," in *www.msoon.com*, November 2012.

[113] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Infocom*, New York, June 2002, pp. 1567–1576.

[114] E.J. Duarte-Melo and M. Liu, "Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks," in *IEEE Global Telecommunications Conference*, Taipei, Taiwan, November 2002, pp. 21–25.

[115] G.P. Halkes and K.G. Langendoen, "Crankshaft: An energy-efficient mac-protocol for dense wireless sensor networks," in *Proc. 24th IEEE International Conference on Distributed Computing Systems*, Tokyo, Japan, March 2004, pp. 590–597.

[116] M.I. Brownfield, T. Nelsonb, S. Midkiff, and N.J. Davis, "Wireless sensor network radio power management and simulation models," *The Open Electrical Electronic Engineering Journal*, vol. 4, pp. 21–31, April 2010.

[117] C. Alippi, G. Anastasi, M. Francesco, and M. Roveri, "Energy management in wireless sensor networks with energy-hungry sensors," *IEEE Instrumentation and Measurement Magazine*, vol. 12, no. 2, pp. 16–23, April 2009.

[118] S. J. Kim, B. Kim, S. Nam, D. Markovic, S.G. Lee, and J. Lee, "Challenges and directions of ultra low energy wireless sensor nodes for biosignal monitoring," *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 986–989, May 2012.

[119] R. Bellman, *A Markovian Decision Process*, vol. 6, Journal of Mathematics and Mechanics, 1957.

[120] *MSP430X1XXX Family, User's Guide*, Texas Instruments, 2006.

CHAPTER 7. BIBLIOGRAPHY

Acronyms

Ack	Acknowledgement
ARQ	Automatic Repeat reQuest
BS	Base Station
CDF	Cumulative Distribution Function
dof	Degrees of freedom
GF	Galois Field
LT	Luby Transform
MAC	Medium Access Control
MDP	Markov Decision Process
Mi	Mica mote
μC	Microcontroller
PDF	Probability Density Function
PHY	Physical
RLNC	Random Linear Network Coding
RSSI	Received Signal Strength Indicator
SPI	Serial Peripheral Interface
T	TelosB mote
TAQR	TDMA with ARQ
TDMA	Time Division Multiple Access
TNC	TDMA with Network Coding
W	Waspmote mote
WSN	Wireless Sensor Network
XOR	Exclusive OR
Z	Zolertia mote